МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Оксана ШОВКОПЛЯС
(підпис)
грудня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістр

зі спеціальності 122 Комп'ютерні науки,
освітньо-професійної програми «Інформатика»
на тему: «Інформаційна технологія аналізу продуктивності та зручності використання мов програмування у різних галузях»
здобувача групи ІН.м - 33 Бараненко Олександр Сергійович

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Олександр БАРАНЕНКО
(підпис)

Керівник,
старший викладач,
кандидат фіз.-мат. наук Анна БАДАЛЯН
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Оксана ШОВКОПЛЯС
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістра

зі спеціальності 122 Комп'ютерні науки, освітньо-професійної програми «Інформатика»
здобувач групи ІН.м-33 Бараненко Олександр Сергійович

1. Тема роботи: ««Інформаційна технологія аналізу продуктивності та зручності використання мов програмування у різних галузях»
затверджую наказом по СумДУ від *«03» грудня 2024 року № 1257-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 06 грудня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження. 2) Огляд технологій, що використовуються для аналізу продуктивності та зручності мов. 3) Розробка технології аналізу продуктивності та зручності мов. 4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____
6. Консультанти до проекту (роботи), із зазначенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ Керівник _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, що використовуються для аналізу продуктивності та зручності мов*		
3	*Розробка технології аналізу продуктивності та зручності мов*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 38 стр., 6 рис., 2 додатки, 24 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв’язанню важливої практичної задачі аналізу продуктивності та зручності мов програмування у різних галузях.

Об’єкт дослідження — аналіз продуктивності та зручності мов програмування.

Мета роботи — розробка інформаційної технології аналізу продуктивності та зручності мов програмування на основі тестувань.

Методи дослідження — аналіз літератури та існуючих інструментів, метод тестування, порівняльний аналіз, метод візуалізації даних.

Результати — розроблено платформу на якій продемонстровано результати проведення тестування продуктивності різних мов програмування та є можливість збирання нової інформації за рахунок проходження тестувань зручності.

МОВИ ПРОГРАМУВАННЯ, ПРОДУКТИВНІСТЬ, ЗРУЧНІСТЬ, ТЕСТУВАННЯ, C++, RUST, GO, PYTHON, JAVA SCRIPT, RUBY.

Зміст

ВСТУП

Обґрунтування вибору теми роботи. У сучасному світі програмування є основою для багатьох технічних рішень, які забезпечують роботу систем, які використовуються в різних галузях. Інтерес до обчислювальної ефективності та продуктивності програмного забезпечення тільки збільшується, що сприяє всебічному розвитку мов програмування. У зв’язку з цим, виникає потреба в аналізі та порівнянні мов програмування для обрання найбільш ефективної мови для конкретних задач.

Актуальність. Продуктивність мов програмування зростає з кожним днем, та не завжди можна легко зрозуміти переваги та недоліки різних мов програмування для вирішення виникаючих завдань. Аналіз продуктивності та зручності використання допоможе ефективно обирати потрібні ресурси для вирішення проблем.

Об’єкт дослідження. Процес аналізу продуктивності та зручності використання мов програмування.

Предмет дослідження. Методи та технології оцінки продуктивності та зручності сучасних мов програмування.

Новизна. Дослідження базується на тестуванні продуктивності й аналізі обчислювальних характеристик сучасних мов програмування, таких як C++, Go, Rust, JavaScript, Python та Ruby, з урахуванням їх останньої оптимізації та тенденцій розвитку.

Структура. Дана робота складається зі вступу, аналітичного огляду, порівняння різних мов програмування, програмної реалізації технології оцінювання мов програмування, висновків, списку використаних джерел та додатків.

1 ОГЛЯД МОВ ПРОГРАМУВАННЯ

Мови програмування з самого початку використання потребують хороших спеціалістів для розв’язування все нових та нових задач. Якщо раніше це було полегшення процесу спілкування між людиною та машиною, а основними задачами були автоматизація обчислень, автоматизації та військових та наукових досліджень. Сучасні мови програмування виконують широкий спектр завдань від великих корпоративних систем до невеликих сценаріїв автоматизації, допомагаючи вирішувати потреби практично у всіх сферах діяльності.

На сьогодні у використанні спеціалістів нараховується більше ніж 700 мов програмування, які використовуються для максимально різних задач. В даному огляді будуть представлені одні із самих популярних та використовуваних мов.

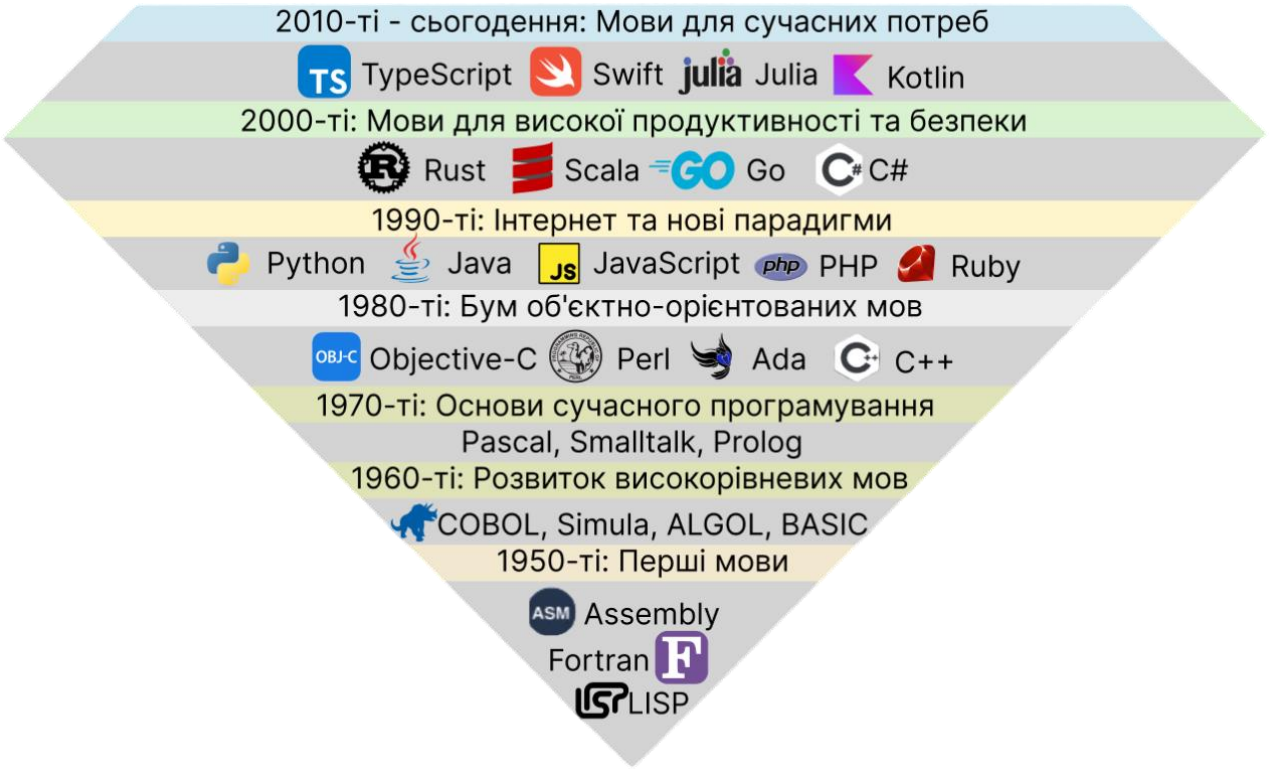

Рисунок 1.1 Зображення розвитку мов програмування

1.1 Мови програмування для високопродуктивних систем

1.1.1 C та C++

Розробники вважають C однією з найшвидших мов програмування для низькорівневої розробки. C – чудовий вибір для низькорівневих додатків, тоді як C++ добре підходить для корпоративного програмного забезпечення. C та C++ мають схожий синтаксис, але C++, як підмножина C, має ширшу функціональність. Використання C має значення для високопродуктивних критично важливих додатків: розробка операційних систем, графічного інтерфейсу користувача (GUI), движків браузерів, вбудованих систем та інших випадків використання [1]. Мова C стала фундаментом завдяки своїм особливостям, а саме тим, що це низькорівнева мова, C надає користувачам значний контроль над апаратним забезпеченням, дозволяючи працювати безпосередньо з пам’яттю. Це робить її ідеальним вибором для системного програмування та розробки драйверів. Синтаксис C відносно простий, що полегшує вивчення. Компільований код C зазвичай дуже ефективний, забезпечуючи високу продуктивність. C використовується для створення операційних систем (Linux, Unix), вбудованих систем (мікроконтролери), компіляторів та багатьох інших програм.

Мова C++ це еволюція C з об’єктно-орієнтованим програмуванням. C++ розширила C, додавши підтримку об'єктно-орієнтованого програмування (ООП). Це дозволяє створювати більш структуровані та повторно використовувані програми. C++ надає більше абстракцій, ніж C, що може спростити розробку складних систем. C++ дозволяє програмістам обирати між процедурним і об'єктно-орієнтованим стилями програмування. C++ використовується в широкому спектрі областей, включаючи розробку ігор, наукові обчислення, системне програмування та створення графічних інтерфейсів [2].

Мову C використовують частіше за все для: низькорівневої розробки, ігрових додатків, високопродуктивних серверів, використання у фінансовій індустрії, наукових обчисленнях та розробки додатків у реальному часі.

Мову C++ використовують для ігрової індустрії, фінансових установ, наукових обчислень, додатків у реальному часі, високопродуктивних серверів, спеціальних драйверів та низькорівневих системних компонентів.

Таблиця 1.1 Відмінності C та C++

Характеристика	C	C++
Парадигма програмування	Процедурна	Об'єктно-орієнтована, процедурна
Абстракція	Нижчий рівень	Вищий рівень
Орієнтація на дані	Менша	Більша
Шаблони	Немає	Є
Керування пам'яттю	Ручне	Ручне, але з можливостями автоматичного керування

1.1.2 Rust

Rust це сучасна мова відома своєю продуктивністю та безпекою. Rust поєднує швидкодію системного програмування з надійними механізмами захисту від помилок. Продуктивність мови забезпечується компіляцією в машинний код, що дозволяє отримати швидкий, ефективний код, подібний до C і C++. У Rust немає runtime, а збірка сміття не потрібна, що робить його ідеальним для розробки системних додатків, зокрема, операційних систем, серверного програмного забезпечення, ігрових рушіїв тощо. Продуктивність часто високо оцінюється в багатопотокових та ресурсозатратних програмах. Однією з головних переваг Rust є його сувора модель керування пам’яттю без використання збирача сміття (garbage collection). Rust забезпечує це через систему запозичень (borrow checker), яка контролює доступ до ресурсів у момент компіляції, забезпечуючи безпеку і унеможливлюючи типові помилки з пам’яттю, такі як use-after-free або нульові вказівники. Це дозволяє уникати багатьох помилок, властивих C/C++, зберігаючи при цьому ефективність роботи з пам'яттю [3].

Попри переваги, Rust має доволі високий поріг входження. Система запозичень та механізм керування пам’яттю вимагають від розробника

точного розуміння життєвого циклу об’єктів і як вони використовуються. Це може бути складним для новачків, особливо для тих, хто звик до мов з автоматичним керування пам’яттю (наприклад, Python чи Java). Однак, ті, хто освоїв Rust, зазвичай відзначають, що мова сприяє кращим практикам розробки, що знижує кількість помилок і сприяє розробці надійних і стабільних програм. Rust забезпечує високий рівень продуктивності та надійності завдяки своїй моделі керування пам’яттю, що захищає від класичних помилок, властивих низькорівневому програмуванню. Хоча вивчення Rust може бути складним, зусилля компенсуються можливістю створювати безпечний і високопродуктивний код [4].

Мову Rust часто використовують для бекенд веб-розробки, індустрії інтернету речей, для розробки компонентів операційних систем, в розроблені драйверів, веб-серверів, мережевих додатків та систем які функціонують у реальному часі.

1.1.3 Go (Golang)

Go (Golang) мова програмування розроблена компанією Google, яка акцентує увагу на простоті використання, підтримці паралелізму та продуктивності. Go був створений з метою забезпечити простоту і швидкість програмування. Однією з головних переваг мови є її мінімалістичний синтаксис, що робить її легкою для вивчення, навіть для тих, хто не має глибоких знань в програмуванні. Відсутність складних конструкцій, таких як класична система наслідування в об'єктно-орієнтованому програмуванні, спрощує розуміння та написання коду. Також має невелику стандартну бібліотеку, що дозволяє швидко почати роботу над проєктами. Завдяки цьому мова дозволяє зосередитись на логіці програми, а не на зануренні в складну теорію [5].

Go має вбудовану підтримку паралелізму через концепцію горутин (goroutines). Горутині — це легкі потоки виконання, які дозволяють ефективно

масштабувати програми, працюючи з багатьма задачами одночасно. Вони набагато легші, ніж традиційні потоки в інших мовах, і керуються планувальником Go. Також надає механізм каналів (channels) для синхронізації між горутинами, що робить роботу з паралельними і конкурентними операціями більш природною та безпечною. Завдяки цьому Go ідеально підходить для розробки високонавантажених серверних застосунків, розподілених систем і веб-сервісів [6].

Go забезпечує високу продуктивність завдяки тому, що компілюється в машинний код. Завдяки простій моделі пам’яті та автоматичному управлінню пам'яттю (через збірник сміття), Go може виконувати програми на високому рівні швидкості, близькому до таких мов, як C чи C++. Однак, через наявність збирача сміття (garbage collector), Go може мати деяке навантаження на продуктивність у ситуаціях, де керування пам’яттю є критичним. Тим не менш, Go швидко зростає за популярністю в таких областях, як створення веб-сервісів і мікросервісів, завдяки своїй ефективності і спрощеній роботі з паралелізмом.

Go частіше за все використовують для розробки ПО для веб серверів, хмарних додатків, мікросервісів, для інструментів контейнеризації такі як Docker, для API, інструментів командного рядка та розробки блокчейн-технології [7].

1.2 Мови програмування для швидкої розробки та прототипування

1.2.1 Python

Python це універсальна мова програмування, що поєднує зручність використання та високу продуктивність завдяки великій екосистемі бібліотек. Він відомий простотою і легкістю у використанні. Його синтаксис лаконічний і легко читається, що робить Python доступним для початківців і зручним для професійних розробників. Завдяки своїй структурі Python дозволяє писати код швидко та ефективно, зводячи до мінімуму зайві конструкції та складні

концепції. Python також має велику стандартну бібліотеку, яка охоплює безліч задач: робота з файлами, мережами, веб-розробкою, базами даних та інше. Це дозволяє вирішувати широкий спектр проблем без необхідності використовувати сторонні інструменти [8].

Python — це універсальна мова, яка використовується в різних сферах, включаючи веб-розробку, машинне навчання, науку про дані, штучний інтелект, автоматизацію завдань, розробку ігор та навіть мобільні додатки. Зручність інтеграції з іншими мовами і технологіями також робить Python популярним вибором для складних, багатокомпонентних проєктів. Ця універсальність забезпечується потужною екосистемою бібліотек і фреймворків, таких як Django і Flask для веб-розробки, TensorFlow і PyTorch для штучного інтелекту, і Kivy для створення мобільних додатків. Завдяки цьому Python підходить для створення різних типів застосунків, дозволяючи розробникам використовувати одну мову для різних задач [9].

Python сам по собі може бути менш продуктивним у порівнянні з такими мовами, як C++ чи Java, особливо у високонавантажених системах. Однак завдяки потужним бібліотекам, таким як NumPy і Pandas, Python здатний забезпечувати високу продуктивність у задачах, пов’язаних з обробкою даних, науковими обчисленнями, машинним навчанням та аналітикою. NumPy дозволяє ефективно працювати з багатовимірними масивами та виконувати складні математичні операції, використовуючи швидкі оптимізовані алгоритми. Бібліотека написана з використанням C, тому її функції можуть виконуватися на рівні продуктивності, близькому до нативного C. Pandas надає інструменти для аналізу та обробки даних у вигляді таблиць, що полегшує роботу з великими обсягами даних і підтримує швидкі операції над ними. Крім цього, Python має бібліотеки для паралельного і розподіленого обчислення, такі як Dask і Ray, що дозволяють масштабувати обробку даних і збільшувати продуктивність в масштабних проєктах [10].

1.2.2 JavaScript

JavaScript це одна з основних мов для веб-розробки, яка виділяється своєю гнучкістю, продуктивністю у браузері та легкістю навчання. Це ключова мова для інтерактивних веб-додатків і єдина мова, яка може працювати безпосередньо у браузері. JavaScript дозволяє створювати динамічні елементи на веб-сторінках, такі як інтерактивні форми, анімації, сповіщення, а також повноцінні SPA (single-page applications) без перезавантаження сторінки. JavaScript легко інтегрується з HTML і CSS, а також підтримується широким набором фреймворків і бібліотек, таких як React, Angular та Vue, що дозволяє швидко створювати складні та зручні у використанні інтерфейси. Крім того, JavaScript разом із Node.js відкрив можливості для розробки серверної частини додатків, зробивши його універсальним інструментом як для фронтенд-, так і для бекенд-розробки [11].

JavaScript має високу продуктивність завдяки сучасним рушіям JavaScript (як-от V8 від Google), які активно оптимізуються для роботи у браузерах. Ці рушії виконують JIT-компіляцію (just-in-time compilation), що дозволяє перетворювати JavaScript-код у машинний код безпосередньо під час виконання, забезпечуючи швидкодію навіть для складних веб-додатків. JavaScript також підтримує асинхронне програмування, яке дозволяє ефективно обробляти запити до сервера, анімації та інші ресурсоємні задачі, не блокуючи роботу інтерфейсу користувача. Це особливо важливо для веб-додатків, де одночасна обробка великої кількості запитів і оновлення інтерфейсу забезпечують плавність і швидкодію веб-сторінок [12].

JavaScript має синтаксис, який легко освоїти, навіть без досвіду в програмуванні. Багато концепцій JavaScript є зрозумілими для новачків, і завдяки простому, зручному підходу до роботи з веб-сторінками та елементами DOM, JavaScript ідеально підходить для тих, хто хоче швидко створювати веб-додатки. Має велику спільноту розробників, численні навчальні матеріали, курси, а також доступ до безлічі інструментів і

фреймворків, що робить його легко доступним для навчання. Це одна з найпоширеніших мов програмування, тому новачки мають доступ до великої кількості прикладів, форумів і документації [13].

1.2.3 Rudy

Ruby це мова програмування, яка цінується за свій простий і лаконічний синтаксис, що дозволяє писати зрозумілий та читабельний код. Її основне застосування в веб-розробці пов’язане з популярним фреймворком Ruby on Rails.

Ruby відомий своєю «людяністю» — він був створений, щоб бути зрозумілим і зручним для програмістів. Синтаксис Ruby близький до природної мови, що дозволяє розробникам легко писати код, орієнтований на розв’язання задач, а не на технічні деталі. Завдяки цьому програмісти можуть швидко створювати й тестувати додатки, а сам код виходить зрозумілим і легким для підтримки. Ruby також є об’єктно-орієнтованою мовою, що спрощує роботу з модульним і повторюваним кодом.

Набув популярності завдяки фреймворку Ruby on Rails — потужному інструменту для швидкої розробки веб-додатків. Rails пропонує багату структуру і стандарти для створення веб-застосунків, що дозволяє розробникам фокусуватися на логіці програми замість налаштувань і конфігурацій. Rails підтримує модель MVC (Model-View-Controller), яка дозволяє легко організувати структуру коду та зробити додаток більш зрозумілим і керованим.[14]

Rails має багато вбудованих функцій, таких як генератори коду, робота з базами даних, маршрутизація, аутентифікація користувачів, що значно пришвидшує розробку та впровадження додатків. Це робить Ruby on Rails популярним вибором для стартапів і компаній, які хочуть швидко вивести свій продукт на ринок.

З точки зору продуктивності, Ruby поступається багатьом мовам, особливо тим, що компілюються (як-от Java або Go). Продуктивність Ruby обмежена його інтерпретованою природою, що робить його менш ефективним для задач, де потрібна висока швидкість і ресурсоємність. Проте Rails оптимізований для підвищення продуктивності веб-додатків, завдяки кешуванню, ефективній роботі з базами даних та розширенням, таким як Active Record, що дозволяє мінімізувати час обробки запитів. Для високонавантажених додатків, де критична продуктивність, Ruby може вимагати додаткових рішень (наприклад, використання кешування, індексації баз даних або переходу на інші високопродуктивні мови для окремих компонентів) [15].

Якщо розглядати список вище зазначених тем як топ, то можна сказати що. C++ посідає перше місце завдяки низькорівневому доступу до пам’яті та мінімальним накладним витратам під час виконання, що робить її ідеальним вибором для систем реального часу, ігор і додатків, критично важливих для продуктивності. Мова забезпечує точний контроль над ресурсами, дозволяючи досягати максимальної ефективності, особливо у великих проєктах.

Rust займає друге місце через акцент на безпеці, керування пам’яттю та підтримку паралелізму без збирача сміття, що робить її відмінним вибором для створення безпечного і надійного програмного забезпечення. Rust чудово підходить для веб-серверів, IoT-рішень, мережевих додатків і системного програмування, де важлива безпека й ефективне керування пам’яттю [16].

Go посідає третє місце завдяки ефективній підтримці паралелізму через горутини та канали, які значно спрощують роботу з паралельними задачами. Go популярний серед розробників для створення розподілених систем, веб-сервісів, хмарних додатків, контейнеризації, мікросервісів і блокчейн-технологій, що вимагають високої продуктивності та легкості підтримки [17].

Python вважається однією з найпопулярніших і найлегших для освоєння мов, завдяки своєму простому синтаксису та великій кількості доступних бібліотек. Він ідеально підходить для швидкого прототипування, наукових

досліджень, аналізу даних, розробки веб-додатків і особливо для штучного інтелекту та машинного навчання. Бібліотеки, такі як NumPy, PyTorch, TensorFlow і Pandas, роблять Python лідером у галузях обробки великих обсягів даних і створення складних AI-моделей. Хоча Python є інтерпретованою мовою і поступається за швидкістю мовам, що компілюються, використання JIT-компіляторів, як-от PyPy, і розширень на C++ (наприклад, Cython) може суттєво підвищити продуктивність для критичних завдань [18].

JavaScript – це мова, спеціально розроблена для веб-розробки, і на сьогодні вона є невід'ємною частиною сучасного інтернету. JavaScript забезпечує інтерактивність на стороні клієнта і виконується безпосередньо у браузері, що робить її незамінною для створення динамічних веб-сторінок та односторінкових додатків (SPA). Сьогодні, завдяки фреймворкам та середовищам, як-от Node.js, JavaScript також успішно використовується для серверної розробки, що дозволяє реалізувати повноцінний стек для веб-додатків на одній мові. JavaScript підтримується вбудованими JIT-компіляторами, як V8, що значно підвищують її продуктивність, особливо у веб-браузерах.

Ruby – це мова програмування, яка цінується за простоту написання коду та орієнтованість на високу продуктивність розробників. Завдяки своєму читабельному синтаксису Ruby дозволяє розробникам писати ефективний код швидко та з меншими зусиллями, що сприяє швидкому впровадженню змін та зниженню рівня помилок. Ruby широко використовується у веб-розробці завдяки фреймворку Ruby on Rails, який спрощує створення серверних додатків та API. Ця мова зазвичай не призначена для важких обчислень і не так оптимізована на рівні виконання, як компільовані мови, проте Ruby ідеально підходить для малих та середніх проєктів, зокрема стартапів, які потребують швидкої розробки та гнучкості.[19].

2 ПОРІВНЯННЯ ПРОДУКТИВНОСТІ ТА ЗРУЧНОСТІ ВИКОРИСТАННЯ ДЛЯ МОВ ПРОГРАМУВАННЯ

2.1 Python vs C++

Python і C++ це дві потужні мови програмування, кожна з яких має свої переваги та недоліки, залежно від задачі. Python відомий своєю зручністю у використанні, універсальністю та багатою екосистемою бібліотек, особливо в аналізі даних, науці та веб-розробці. C++, у свою чергу, виділяється високою продуктивністю, ефективним керуванням пам’яттю та здатністю працювати з низькорівневими компонентами системи

Переваги та недоліки Python і C++ у різних випадках використання

1. Продуктивність: Python менш продуктивний у порівнянні з C++ через інтерпретовану природу та збирач сміття, який додає затримки в управлінні пам’яттю. Однак завдяки бібліотекам (наприклад, NumPy) Python може обробляти наукові розрахунки швидше за рахунок використання оптимізованого коду на C/C++ під капотом.

C++ забезпечує високу швидкість виконання завдяки компіляції в машинний код і можливості оптимізації пам'яті. Це ідеальний вибір для програм, де потрібна максимально висока продуктивність, наприклад, ігор, операційних систем і програм реального часу.

2. Зручність використання: python має зрозумілий і простий синтаксис, що робить його зручним для швидкого прототипування і розробки. Python широко використовується для створення скриптів, веб-додатків, аналізу даних і машинного навчання.

C++ складніший у вивченні, оскільки має більш складний синтаксис, вимагає точного керування пам’яттю та розуміння таких концепцій, як вказівники, але надає більше контролю і можливостей для оптимізації.

3. Застосування: python часто застосовується в науці про дані, аналізі даних, машинному навчанні, автоматизації та веб-розробці [20].

C++ використовується в розробці ігор, системного програмного забезпечення, графічних рушіях, реальних часах та інших сферах, де критичні швидкість та оптимізація ресурсів.

4. Керування пам’яттю: python має автоматичний збирач сміття, що спрощує керування пам’яттю, але це може викликати затримки в програмах, що потребують високої продуктивності.

C++ дає розробнику повний контроль над керуванням пам’яттю, що дозволяє оптимізувати використання пам’яті, але вимагає більше уваги до запобігання помилок, таких як витік пам’яті.

Таблиця 2.1 Порівняння Python та C++

Критерії	Python	C++
Продуктивність	Низька, але може підвищуватись завдяки бібліотекам (наприклад, NumPy)	Висока, завдяки компіляції та оптимізації
Зручність використання	Простий, зрозумілий синтаксис, швидке прототипування	Складний синтаксис, але гнучкість і контроль
Керування пам’яттю	Автоматичний збирач сміття	Повний контроль, можливість оптимізації
Універсальність	Універсальна мова для науки, веб, ML	Переважно для високопродуктивних програм
Сфера застосування	Наука про дані, автоматизація, веб-розробка	Ігри, графіка, системне програмування
Поріг входження	Низький	Високий, вимагає розуміння вказівників та керування ресурсами
Бібліотеки	Велика екосистема для ML, науки, автоматизації	Є бібліотеки для обчислень, але їх складніше інтегрувати
Швидкість розробки	Висока, завдяки простому синтаксису та динамічній типізації	Нижча, потребує компіляції та налаштування пам'яті

2.2 Java vs JavaScript

Java і JavaScript — це дві абсолютно різні мови програмування, які часто плутають через схожість у назві. Однак вони мають різне призначення, синтаксис, екосистеми та використовуються для різних завдань, особливо у веб- і мобільній розробці.

Порівняння Java і JavaScript для веб- і мобільного розроблення

1. Застосування у веб-розробці java використовується переважно для серверної розробки (бекенд) веб-застосунків. Веб-фреймворки на основі Java, такі як Spring, Hibernate та JSF (JavaServer Faces), дозволяють створювати високонавантажені серверні частини веб-додатків. Java кодується, компілюється, а потім виконується на сервері, обробляючи запити і відповідаючи клієнтським браузерам.

JavaScript: основна мова для фронтенд-розробки, яка виконується безпосередньо у браузері та забезпечує інтерактивність на веб-сторінках. За допомогою JavaScript можна створювати динамічний контент, анімації, обробляти події і змінювати елементи HTML у реальному часі. Крім того, завдяки Node.js, JavaScript також активно використовується для серверної розробки, що дозволяє використовувати одну мову як для фронтенду, так і для бекенду [21].

2. Застосування у мобільній розробці java сновна мова для розробки Android-додатків. Офіційний фреймворк Android SDK надає розробникам інструменти для створення нативних додатків на Java, що забезпечує високу продуктивність та інтеграцію з функціями Android. Додатки на Java для Android добре оптимізовані та забезпечують повний доступ до функцій пристрою, таких як камери, датчики та GPS.

JavaScript: використовується для мобільної розробки переважно за допомогою кросплатформених фреймворків, таких як React Native, Ionic, і PhoneGap. Зазвичай JavaScript-код транслюється у нативні компоненти, що дозволяє створювати додатки для iOS та Android одночасно. Це забезпечує швидший процес розробки і зниження витрат, однак кросплатформені додатки можуть бути менш продуктивними та не завжди підтримують доступ до всіх нативних функцій.

3. Продуктивність: java продуктивніша в порівнянні з JavaScript для обчислювально інтенсивних завдань, особливо в мобільних додатках. Будучи компільованою мовою, Java кодується у байт-код, який потім виконується на

Java Virtual Machine (JVM), що дозволяє досягати високої швидкодії та оптимізації ресурсів.

JavaScript: працює повільніше за Java в сценаріях, де потрібні ресурсоємні обчислення. Продуктивність може покращуватись завдяки сучасним JavaScript-рушіям (наприклад, V8), але вона все ж поступається компільованим мовам, особливо для мобільних додатків і бекенд-обчислень.

4. Зручність розробки

Java: вимагає більш складної настройки для мобільної розробки та веб-розробки на серверній частині, проте має чітку структуру, строгий синтаксис і дотримується об’єктно-орієнтованих принципів, що полегшує створення масштабованих додатків. Веб-додатки на Java часто більш складні в налаштуванні, але забезпечують високу продуктивність і надійність.

JavaScript: легша для старту у веб-розробці, особливо на фронтенді. JavaScript має простіший синтаксис і велику кількість бібліотек та фреймворків, що значно пришвидшують розробку. Завдяки широкій підтримці на фронтенді і серверній частині (завдяки Node.js), JavaScript підходить для створення повноцінних веб-додатків із сучасними інтерактивними інтерфейсами.

5. Стабільність і підтримка спільноти

Java: має довгу історію та велику спільноту професійних розробників. З огляду на підтримку великими корпораціями (наприклад, Oracle) і стабільність самої мови, Java є надійним вибором для створення великих, довготривалих проєктів. Android-додатки на Java також мають велику підтримку з боку спільноти.

JavaScript: активно розвивається спільнотою і постійно оновлюється. Спільнота JavaScript — одна з найбільших у світі програмування, а бібліотеки та фреймворки, такі як React і Angular, забезпечують його популярність серед веб-розробників. Однак через швидкі зміни і поява нових фреймворків JavaScript може вимагати частого оновлення знань.[22]

Таблиця 2.2 Порівняння Java та JavaScript

Критерії	Java	JavaScript
Веб-розробка	Серверна частина (бекенд)	Фронтенд і бекеннд (з Node.js)
Мобільна розробка	Нативна андроїд-розробка	Кросплатформенна розробка (React, Native, Ionic)
Продуктивність	Висока, компільована мова	Сердня, інтерпретована мова
Зручність розробки	Більш складне налаштування, суворий синтакси	Простіша для старту, простий синтаксис
Стабільність	Стабільна і перевірена мова	Швидкі зміни, простий розвиток фреймворків
Спільнота	Велика професійна спільнота	Одна з найбільших у світі, активний розвиток

2.3 Rust vs Go

Rust і Go — це дві сучасні мови програмування, кожна з яких пропонує унікальні переваги і підходить для різних типів завдань. Rust акцентує увагу на безпеці та контролі над пам’яттю, забезпечуючи високу продуктивність. Go, у свою чергу, відзначається простотою синтаксису та легкою підтримкою паралелізму, що робить його популярним у розробці масштабованих серверних систем.

Вибір між Rust і Go: порівняння в контексті безпеки, швидкості, простоти та паралельності

1. Безпека та керування пам’яттю

Rust: має систему керування пам’яттю без збирача сміття (garbage collector), яка дозволяє уникати багатьох проблем, пов’язаних з помилками пам’яті, завдяки системі позичання і життєвого циклу змінних (borrow checker). Rust пропонує детальний контроль за керуванням пам'яттю, що

дозволяє запобігти небезпечним ситуаціям, як-от використання вказівників на видалені об’єкти. Це робить Rust ідеальним вибором для системного програмування, де критична безпека і відсутність витоків пам’яті [23].

Go: має збирач сміття, що значно спрощує роботу з пам'яттю для розробників, оскільки їм не потрібно вручну слідкувати за життєвим циклом об’єктів. Це дає змогу уникнути багатьох потенційних помилок, які можуть виникнути при управлінні пам'яттю, але також додає невеликі накладні витрати на продуктивність. Go підходить для ситуацій, де важлива швидкість розробки і менше критичних вимог до контролю за пам'яттю.

2. Продуктивність і швидкість

Rust: завдяки відсутності збирача сміття і низькорівневому контролю над пам’яттю Rust дозволяє досягати продуктивності, близької до C/C++. Це робить Rust відмінним вибором для високопродуктивних додатків, системного програмного забезпечення і проєктів, де критична швидкість виконання (наприклад, ігри, графіка, блокчейн-додатки).

Go: хоча Go не забезпечує такої ж продуктивності, як Rust, він все ще залишається дуже швидким завдяки компіляції в машинний код і оптимізаціям Go-рушія. Go підходить для додатків, де критична обробка великої кількості одночасних запитів, а продуктивність не повинна перевищувати певний рівень. Go добре підходить для серверних додатків, де потрібен баланс між швидкістю виконання і простотою обслуговування.

3. Простота використання

Rust: відносно складна мова з крутим порогом входження, особливо через концепцію позичання (borrow checker) та строгий контроль над керуванням пам’яттю. Rust вимагає від розробника точного розуміння життєвого циклу змінних, що може затримувати початківців, але водночас допомагає уникати помилок у коді. Це робить Rust менш зручним для швидкого прототипування, але чудовим вибором для розробки надійних і безпечних додатків.

Go: має простий, лаконічний синтаксис і легкий поріг входження, що робить його популярним вибором для швидкого прототипування і розробки програм. Go цілеспрямовано уникає складних концепцій (наприклад, наслідування), що робить код легко читабельним і зрозумілим. Go створений для продуктивності розробників, і його простий синтаксис дозволяє швидко навчитися писати ефективний код [24].

4. Підтримка паралельності

Rust: підтримує паралелізм і потокове програмування, але це потребує від розробника більше роботи, щоб забезпечити безпеку в паралельних сценаріях. Rust використовує систему позичання для запобігання гонок даних, але це додає певної складності при написанні багатопотокових програм. Rust надає бібліотеки для роботи з потоками, що дозволяє досягати високої продуктивності, але розробнику необхідно детально розуміти, як забезпечити безпеку при паралельних операціях.

Go: відомий простою підтримкою паралелізму через горутини — легковагові потоки, які автоматично керуються Go-рушієм. За допомогою ключового слова go можна запускати функції асинхронно, а канали (channels) дозволяють легко обмінюватися даними між горутинами. Це робить Go популярним вибором для створення серверів, які обробляють велику кількість одночасних запитів, і додатків, що потребують високої масштабованості.

Таблиця 2.3 Порівняння Rust та Go

Критерії	Rust	Go
Безпека керування пам’яттю	Висока, без збирача сміття	Середня, автоматичний збирач сміття
Продуктивність	Дуже висока, близька до C/C++	Висока, але поступається Rust
Простота використання	Відносно складна для вивчення	Легка у вивченні з простим синтаксисом
Паралельність	Підтримка потоків, але вимагає увагу до безпеки	Простий паралелізм через горутини

Продовження таблиці 2.3

Сфера застосування	Системне програмування, високопродуктивні додатки	Сервери, мережеві застосунки, веб-сервіси
Поріг входження	Високий, складний синтаксис	Низький лаконічний синтаксис
Підтримка спільноти	Зростає, сильна підтримка з боку Mozilla	Висока, підтримка з боку Google

Розглядаючи порівняння різних мов програмування, можна дійти висновку, що вибір мови значно залежить від конкретних вимог до продуктивності, зручності використання, безпеки та специфіки застосування. Мови, що компілюються, такі як C++, Rust і Go, забезпечують високу швидкодію та ефективне керування ресурсами, що робить їх придатними для завдань, де ключовими є продуктивність і паралелізм, наприклад у системному програмуванні, обробці великих обсягів даних і роботі з інфраструктурою.

Водночас, інтерпретовані мови на кшталт Python, JavaScript і Ruby пропонують більшу зручність використання, простоту вивчення та потужну екосистему бібліотек і фреймворків. Це робить їх популярним вибором для наукових обчислень, веб-розробки, створення AI/ML рішень та швидкого прототипування. Кожна з мов має свої сильні та слабкі сторони, що слід враховувати при виборі мови для конкретного проєкту. Тому, важливо ретельно аналізувати завдання, які потрібно вирішити, та підбирати мову, що найкраще відповідає конкретним потребам і технічним вимогам.

2.4 Оцінка зручності мов програмування

Зручність використання мови програмування визначає, наскільки просто або складно розробникам працювати з нею, вивчати її, читати і розуміти код, використовувати стандартні функції та бібліотеки.

Основними факторами використання мов програмування є: синтаксис те легкість навчання, інструменти та екосистема, документація та підтримка спільноти та читабельність коду.

Першою на черзі буде C++ дана мова характеризується складним синтаксисом та високим порогом входу через низькорівневі концепції та керування пам’яттю, представлена велика кількість інструментів та багата стандартна бібліотека, існує велика кількість книг, навчальних матеріалів та стандартизованої документації, читабельність коду стане швидкою тільки з надбаним досвідом тому мова складна для початківців.

Наступна Rust, вирізняється складним синтаксисом суворий контроль пам’яті робить поріг входження високим але код в той же час гарантує безпеку, існує потужна система пакетів Cargo, доволі активна спільнота, корисна документація та навчальні матеріали від команди розробників, код читабельний але вимагає розуміння концепції системної безпеки.

Go прості та зрозумілі конструкції, легкий для навчання та користування, доволі розвинена екосистема для веб та серверних додатків, вбудовані інструменти та підтримка IDE, активна спільнота, хороша документація та офіційні посібники від розробників, код вирізняється читабельністю та зрозумілістю завдяки простому синтаксису що легко підтримувати.

Python простий синтаксис, надзвичайно легкий для навчання та особливо для новачків, розвинена екосистема бібліотек та пакетний менеджер pip, спільноту можна назвати найбільшою з представлених мов, велика кількість безкоштовних ресурсів та документації, код вирізняється високою читабельністю та чіткою структурою особливо з PEP8 стандартами.

Ruby простий та лаконічний синтаксис даної мови робить її легким для навчання, існує велика екосистема для веб-розробників, активна спільнота велика кількість ресурсів та підтримка особлива для Rails, код можна назвати лаконічним та зрозумілим особливо для веб-застосунків.

JavaScript основні синтаксиси можна назвати простими але асинхронні конструкції можуть бути складними для новачків, широка екосистема та підтримка багатьох IDE, найбільша спільнота серед мов для веб-розробки, безліч документації та навчальних матеріалів, читабельність коду за умови використання сучасних стандартів ES6+ але залежить від стилю кодування.

Далі представлено топ 10 самих популярних мов програмування викладених в IEEE Spectrum Ranking які компонують топ на основі таких джерел як GitHub, Twitter, TIOBE, Stack Overflow, Reddit, і Google.

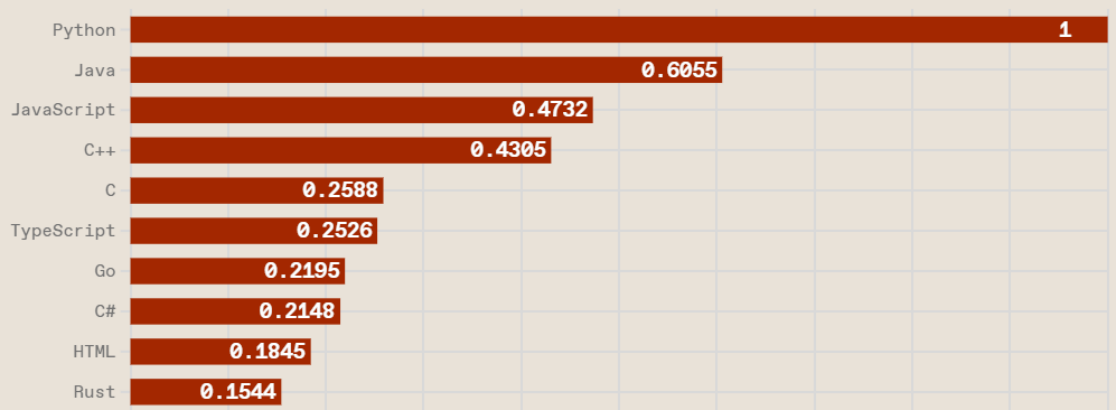

Рисунок 4.1 Топ 10 мов за даними IEEE Spectrum Ranking

Як можна побачити 5 із 6 мов проаналізованих в даному звіті входять до топ 10 мов програмування за популярністю але тренди змінюються та ніхто не знає яка мова посяде топ-1 в наступному році.

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТЕХНОЛОГІЇ ОЦІНЮВАННЯ

3.1 Проведення тестів продуктивності

Для проведення тестів продуктивності для різних мов програмування було обрано три задачі а саме: знаходження числа Фібоначчі, сортування та множення матриць. Цей вибір обґрунтований рядом факторів що дозволять оцінити ефективність мов програмування в різних галузях обчислення а саме обробки даних та маніпулюванням великим обсягом інформації.

Знаходження числа Фібоначчі є класичним тестом для оцінки продуктивності в контексті рекурсивних обчислень та використання підходів для оптимізації. Це завдання дозволяє порівняти продуктивність при рекурсивних обчисленнях та продуктивність при великих значеннях на вході.

Простота рекурсивної реалізації алгоритму обчислення чисел Фібоначчі може бути дуже неефективною через дублювання обчислення. Однак при оптимізації наприклад у випадку використання мемоізації або динамічного програмування, продуктивність може бути значно покращеною. Цей аспект дозволяє оцінити наскільки швидко мови програмування можуть виконувати обчислення в контексті неефективних та оптимізованих рішень.

Обчислення при великих значеннях на вході добре підходять для вимірювання часу виконання при збільшенні вхідного значення, оскільки при зростанні значення числі відбувається збільшення кількості обчислень.

Для проведення тестів продуктивності а саме знаходження числа Фібоначчі для кожної мови було написано код тестування. В якому було виконано рекурсивний метод.

Було проведено 10 тестів для кожної мови в кожному тесті було змінено значення для якого потрібно знаходити число Фібоначчі. Результати тестів приведені в секундах та було виокремлено середнє значення виконання коду програми.

Таблиця 3.1 Результати тестування мов програмування на визначення числа Фібоначчі

	C++	Rust	Go	Python	JavaScript	Ruby
N=5	0,00000025	0,00000271	0000000027	0,000003338	0,000069	0,0000293
N=10	0,00000116	0,00000309	0,000000088	0,00001502	0,000071	0,0000856
N=15	0,00000557	0,00000776	0000000425	0,047351	0,000235	0,0007726
N=20	0,00005595	0,00010188	0,000006334	0,0928	0,001078	0,000946
N=25	0,00060032	0,00110093	0,000103826	0,0146	0,005333	0,008083
N=30	0,00594012	0,12535739	0,005456309	0,4821	0,011425	0,10199
N=35	0,0632489	0,13207521	0,082751244	7,497	0,148604	1,076
N=40	0,837034	1,335	0,812638107	54,111	1,366	11,71
N=45	8,40	15,15	7,595	189,20	15,7	133,69
N=50	100,14	311,53	90,31	563,4	125,3	394,21
Середнє значення	10,99	46,35	9,88	82,28	14,85	54,27

Проаналізувавши результати проведення тестів можна зробити наступні висновки:

1. Мова Go продемонструвала самий швидкий час виконання серед усіх мов з часом 9,88 секунд. Ця мова має компілятор який оптимізований для багатопоточності і включає в себе вбудоване керування пам’яттю, що дозволяє ефективно розподіляти ресурси для задач. Крім цього ця мова підтримує високий рівень оптимізації при компіляції, що дозволяє досігати швидкого виконання навіть у порівнянні з низькорівневими мовами.

2. Мова C++ продемонструвала другий за швидкістю результат з часом 10,99 секунд. C++ як одна з найстаріших та найоптимізованіших мов для високопродуктивних обчислень, вона надає користувачу гнучкість в управлінні пам’яттю власноруч, що знижує накладні витрати що в свою чергу дозволяє досягти високою продуктивності, особливо для ресурсомістких задач.

3. Java Script мова яка займає третє по швидкості місце з часом 14,85 секунд, показує що сучасні браузери та інтерпретатори дозволяють

виконувати обчислення з високою швидкістю. Хоча сама мова найчастіше використовується в браузерах для роботи з інтерфейсом, її продуктивність завдяки оптимізації дозволяє ефективно виконувати обчислення не сильно відстаючи від других мов які більше для цього призначені. Це робить Java Script хорошим вибором у випадку роботи з важкими веб застосунками.

4. Rust виявилась більш повільною чим попередні мови з часом 46,35 секунд, але все ж досить конкурентоспроможною. Ця мова відома як та що надає високий рівень безпеки при управлінні пам’яттю і продуктивність цієї мови може бути наближена до других мов тільки при належній оптимізації. Rust особливо популярна серед проектів що потребують як продуктивною так і безпечної роботи.

5. Ruby з часом 54,27 продемонструвала повільну продуктивність, що є типовим для інтерпретованих мов. Ця мова вирізняється своєю читабельністю та зручністю але при виконанні забач які потребують важких обчислювальних задач її продуктивність є обмеженою.

6. Мова Python виявилася самою повільною серед представлених з часом 82,28 секунд. Ця мова використовує більше часу та пам’яті на рекурсивні та ресурсомісткі завдання, особливо порівняно з компільованими мовами. Однак це не заважає Python залишати популярним вибором завдяки своїй простоті та широкому спектру бібліотек, що робить її хорошим вибором у задачах де продуктивність не є критичною.

Наступним тестом ефективності мов програмування стала задача на сортування великих масивів даних. Сортування є важливим завданням, яке використовуються у багатьох програмних системах і є типовим прикладом вимірювання продуктивності мов програмування у роботі з масивами даних.

Дане завдання дозволяє оцінити ефективність алгоритмів сортування, порівняння швидкості реалізації таких алгоритмів як швидке сортування, злиття, бульбашкове сортування та інші дає змогу оцінити те як різні мови програмування виконують операції з обробки даних. Тест дозволяє порівняти час сортування при різних обсягах даних. Ця задача дозволяє оцінити

ефективність мов програмування при роботі з різними обсягами даних від невеликих масивів до великих наборів інформації. Мови програмування можуть використовувати різні підходи до оптимізації сортування включаючи вбудовані функції чи бібліотеки для високо продуктивних обчислень. Це дозволить порівняти не лише базову ефективність алгоритмів, а ще і здатність швидко виконувати широко поширені операції з даними.

Таблиця 3.2 Результати тестування мов програмування на сортування великих масивів чисел.

	C++	Rust	Go	Python	JavaScript	Ruby
1..100000	0,1964	0,074526	0,060412	0,125228	0,221781	0,450183
1..200000	0,2066	0,157924	0,071531	0,272608	0,809775	0,698793
1..300000	0,2185	0,221735	0,082452	0,375551	0,611935	1,4944985
1..400000	0,2237	0,307548	0,075245	1,188565	0,643578	1,6113559
1..500000	0,2654	0,394586	0,081286	1,985421	0,999342	2,2334464
1..600000	0,2284	0,469585	0,104563	1,521785	0,736542	2,3061486
1..700000	0,2485	0,614856	0,075305	2,059445	0,864323	4,5538664
1..800000	0,2334	0,700856	0,102546	3,383452	0,892315	5,8574365
1..900000	0,2432	0,733254	0,048135	2,335468	0,862321	3,2786925
1..1000000	0,2502	0,799556	0,133524	1,335246	0,985623	3,9576002
Середній час	0,2314	0,5374	0,0834	1,4587	0,7628	2,6942

Після аналізу даного тестування можна зробити висновки що знову мова Go має самий швидкий результат що вказує на її ефективність при виконанні паралельних операцій, завдяки простоті в використанні горутини для асинхронних обчислень. Але хоча ця мова є високопродуктивною для деяких специфічних обчислень таких як складні обчислення та маніпулювання великими даними можуть знадобитися другі мови.

Мова C++ показала хороший результат що свідчить про ефективність і швидкість виконання операцій завдяки доступу до пам’яті та оптимізованому управлінні ресурсами. Мова Rust на умовному третьому місці що показує достатню конкурентоспроможність це показує те що мова поєднує в собі високу продуктивність з сучасними механізмами безпеки що робить мову

більш універсальною для багатьох задач. Java Script показала досить хороший результат хоча сама мова не сильно підходить для даної задачі. Однак при постійній оптимізації ця мова може біти більш універсальною для багатьох задач.

Python як інтерпретована мова показала значно повільніший результат у порівнянні з компільованими мовами. Хоча мова є досить популярною черед користувачів її продуктивність для задач таких як сортування залишає бажати кращого. Мова Ruby показала найгірший результат, це може бути пов’язано з тим що ця Ruby є мовою високого рівня яка не оптимізована для швидких обчислень.

Останнім тестом продуктивності виступає множення та додавання матриць. Це завдання є більш складним яке часто використовується в числових методах, комп’ютерній графіці, машинному навчанні та інших обчислювальних завданнях. Оскільки ця операція вимагає значних ресурсів вона є хорошим вибором для порівняння продуктивності мов програмування. Цей тест дозволяє оцінити ефективність роботи з великими даними тому що множення великих об’ємів даних є ресурсозатратним процесом який вимірює здатність мови ефективно оброблять та маніпулювати даними.

Також цей тест дозволяє порівняти багатозадачність та паралельність обчислень та подивитися як вони оптимізовані. Мови програмування можуть використовувати різні підходи до оптимізації обчислень, такі як паралельні обчислення та багатопотоковість що дозволяє оцінити їхню здатність до масштабування та швидкості виконання операцій. Ще даний дозволяє оцінити продуктивність на рівні математичних операцій, тому що саме множення матриць тестує ефективність реалізації числових операцій і може бути використаний для вимірювання швидкості виконання складних обчислень, що надзвичайно важливим для наукових та інженерних задач.

Таблиця 3.3 Результати тестування мов програмування при перемноженні та додаванні матриць різного розміру.

	C++	Rust	Go	Python	JavaScript	Ruby
500	0,1281	0,521	0,580521	0,2033	0,6551	13,05433
600	0,2345	1,081	0,44915	1,2825	1,1606	23,16562
700	0,3434	1,542	1,761	2,3629	2,1874	37,57523
800	0,5478	2,804	1,091	3,5148	3,3059	55,83421
900	0,8863	3,462	4,357	3,6769	4,7902	77,0665
1000	1,2259	3,738	6,33	4,8729	9,6552	107,2612
1100	1,5458	4,007	6,173	5,1277	16,8233	139,755
1200	2,1559	4,679	7,145	9,8843	22,7772	197,7839
1300	2,642	6,801	9,391	12,7884	32,0391	251,177
1400	3,196	6,763	9,243	12,8528	25,6325	317,581
Середнє значення	1,295	3,539	4,652	5,656	11,802	122,935

Аналіз результатів даного тесту показав що, C++ показує самий швидкий час по причині того що ця мова оптимізована для таких операцій і має доступ до низькорівневих бібліотек тому час даних тестів має залишатися низьким навіть для більших розмірів матриць. Мова Rust посіла друге місце по швидкості по тій причині що ця мова теж є компільованою вана має додатуові механізми безпеки що можуть призводити до деяких затримок. Мова Go є мовою високого рівня і хоча її швидкодія зазвичай трохи повільніша ніж у C++ та Rust, вона добре підходить для обчислень що використовують багатозадачніть. Python навіть при використанні допоміжних бібліотек не є кращим вибором для подібних задач. Java Script зазвичай не є сильною у виконанні числових операцій. Мова Ruby є однією з найповільніших мов програмування для числових операцій.

Тести було проведено за допомогою онлайн компіляторів з декількох причин а саме:

1. Онлайн компілятори дозволяють запускати код без встановлення програмного забезпечення.

2. Властивості мого ПК можуть вливати на результати проведення тестів.

3.2 Розробка платформи для висвітлення результатів тестів та опитування користувачів

Під час виконання даної роботи було розроблено платформу для інформаційної технології аналізу зручності та ефективності використання мов програмування. Для зручності використання та можливої подальшої модифікації платформа була розроблена за допомогою мов HTML, Java Script та CSS ці три мови були обрані завдяки декільком ключовим перевагам:

1. Ці три мови дуже прості для освоєння, html дозволяє створити структуру веб застосунків за допомогою інтуїтивно зрозумілих тегів, css використовує правила для стилізації та оформлення а java script додає інтерактивні та динамічні елементи такі як обробка подій та зміна контенту в реальному часі.

2. Ці мови є дуже популярними та підтримуються всіма сучасними браузерами що може забезпечити сумісність і доступність. Також для цих мов є величезний вибір ресурсів такі як туторіали, бібліотеки та фреймворки які значно спрощують процес роботи з ними.

3. Крім всього цього за допомогою чудовою інтеграції між собою ці три мови дають можливість робити сторінки динамічними та інтерактивними що значно покращує зрозумілість та зручність для користувача, також їхня чудова взаємодія між собою дозволяє ефективно розробляти та тестувати веб сайти в одному середовищу. І що саме головне ці технології є відкритими та безкоштовними що робить їх доступними для будь-кого бажаючого.

Дана технологія призначена для комплексної оцінки продуктивності та зручності використання мов програмування. Вона включає в себе результати проведених тестів продуктивності та опитування зручності, що дозволяє отримати всебічні дані для порівняння та оцінки мов програмування в різних галузях.

Дана технологія надає можливість порівнювати різні мови програмування за їхньою швидкістю виконання певних обчислень та обробки

інформації. Це важливо для оцінки ефективності алгоритмів тому що різні мови можуть демонструвати різні результати при виконанні одних і тих самих задач. Проведення тестів дозволяє оцінити як кожна мова використовує обчислювальні ресурси що є важливим для вибору мови в залежності від вимог продуктивності. Деякі мови підтримують багатозадачність на вищому рівні ніж інші що дозволяє ефективніше використовувати ресурси при виконанні великих обчислень.

Зручність використання мов програмування це критично важливий фактор для розробників. Вибір мови залежить не тільки від її продуктивності від того на скільки легко та швидко можна писати на цій мові. Тестування зручності в включає питання які характеризують синтаксис мови, читабельність коду та швидкість досягнення результатів при використанні даної мови.

Нижче показана вступна сторінка даної платформи на якій продемонстровані результати проведених тестів ефективності різних мов програмування.

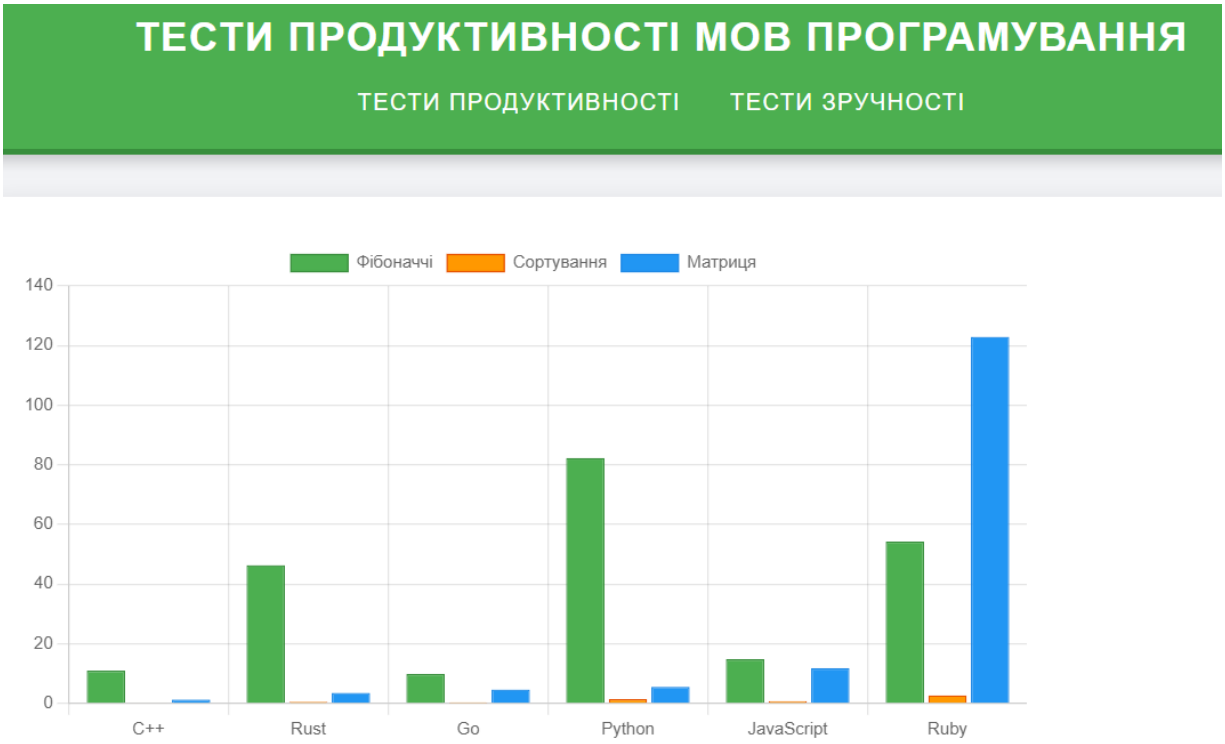

Рисунок 3.1 Вступна сторінка платформи

При необхідності є можливість відсортувати показ графіка для детального дослідження.

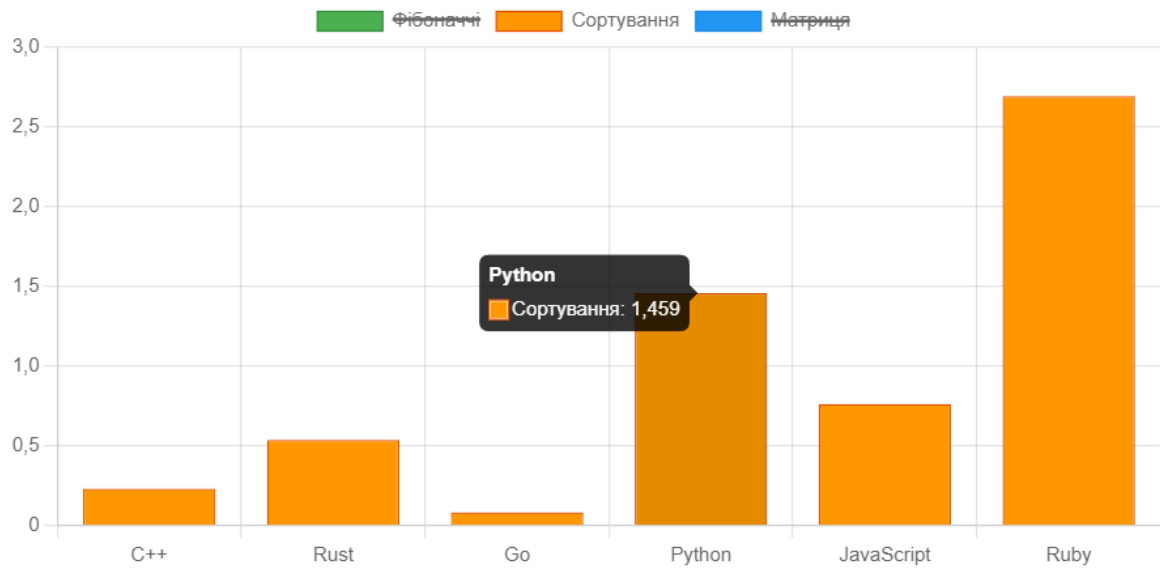

Рисунок 3.2 Можливість сортування

Опитування користувачів дозволяє збирати інформацію про те як розробники оцінюють ці аспекти в реальному часі.

Ця платформа є корисною для широко кола користувачів а саме для розробників, інженерів та дослідників в галузі програмного забезпечення. А саме розробникам платформа дозволить обирати відповідну мову для конкретної задачі з урахуванням як продуктивності так і зручності. Для інженерів це буде корисно для висвітлення і при модернізації проведення тестів ефективності для різних мов програмування. Для дослідників ця платформа може допомогти в порівняння та дослідженні мов програмування в області програмного забезпечення.

Завдяки тому що платформа включає в себе як об’єктивні тести продуктивності так і суб’єктивні відгуки про зручність, вона дозволяє отримати всебічну картину можливостей кожної мови.

Далі представлена сторінка з тестуванням для визначення зручності використання мов програмування.

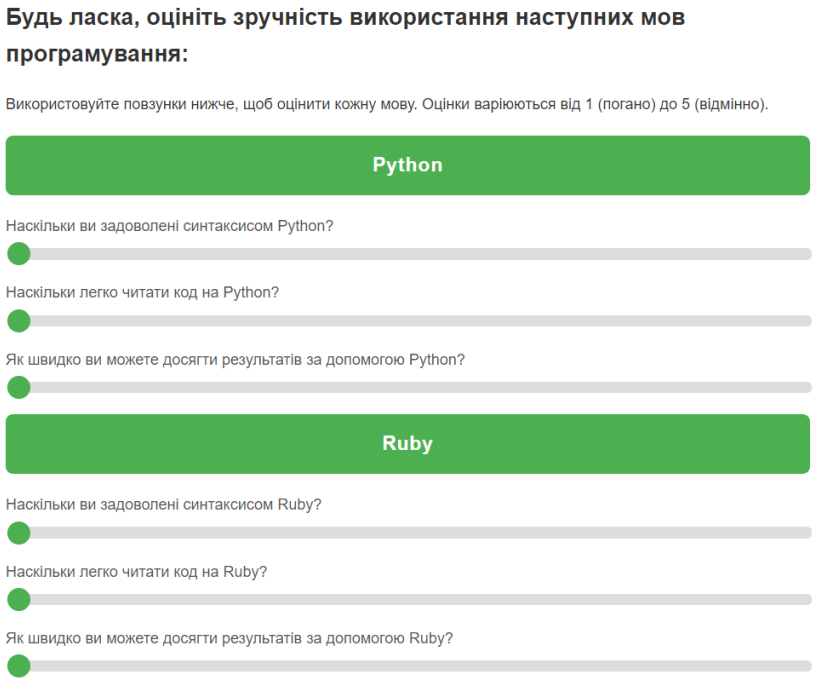

Рисунок 3.3 Опитування для розробників про зручність про зручність

При необхідності можна додавати нові питання різних видів. Після виставлення оцінок натискається кнопка надіслати яка знаходиться в кінці тестування, потім з’являються результати тестування з виставленими оцінками в кожній категорії.

Мова	Синтаксис	Читабельність	Швидкість
Python	3	4	2
Ruby	2	3	4
Go	4	3	2
Rust	3	4	2
JavaScript	2	3	5
C++	5	3	1

Рисунок 3.4 Результати опитування зручності використання.

ВИСНОВКИ

В результаті виконання роботи було розроблено інформаційну систему аналізу продуктивності та зручності мов програмування. Було проведено та реалізовано тести ефективності та зручності.

У вході виконання кваліфікаційної роботи магістра було виконані наступні завдання:

1. Проведено аналіз проблемної області, визначено актуальність створення системи для порівняння мов програмування.

2. Розглянуто сучасні підходи до тестування продуктивності та зручності використання.

3. Обрано системи для розробки системи, включаючи HTML, CSS, JavaScript.

4. Було реалізовано технологію тестування, яка охоплює завдання з обробки даних, складних обчислень і роботу з пам’яттю для кожної мови програмування.

5. Проаналізовано результати тестування, зроблено висновки щодо продуктивності та зручності використання мов програмування.

Надалі планується удосконалення системи реалізації нових можливостей для скорочення часу проведення аналізу продуктивності та отримання зворотного зв’язку щодо зручності використання, що сприяє більш обґрунтованим рішенням для оптимізації у виборі мови для конкретних проектів.

СПИСКИ ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Lincopinis D. et al. C Programming Language: A Review // Journal of Universal Computer Science. 2021. Vol. 27, № 1.

2. Galindo J., Galindo P., Corral J.M.R. Multimedia System for Self-learning C/C++ Programming Language // Learning and Analytics in Intelligent Systems. 2020. Vol. 7.

3. Plauska I., Liutkevičius A., Janavičiūtė A. Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller // Electronics (Switzerland). 2023. Vol. 12, № 1.

4. Bychkov A., Nikolskiy V. Rust Language for Supercomputing Applications // Communications in Computer and Information Science. 2021. Vol. 1510 CCIS.

5. Shrunga G S et al. Study On Go Programming Language // International Journal of Advanced Research in Science, Communication and Technology. 2022.

6. Cox R. et al. The Go programming language and environment // Commun ACM. 2022. Vol. 65, № 5.

7. Daniel Kortschak R. et al. bíogo: a simple high-performance bioinformatics toolkit for the Go language // The Journal of Open Source Software. 2017. Vol. 2, № 10.

8. Шевченко Г. ПЕРЕВАГИ ВИКОРИСТАННЯ ОНЛАЙН-СЕРЕДОВИЩА РОЗРОБКИ «REPLIT» ДЛЯ ВИВЧЕННЯ МОВИ ПРОГРАМУВАННЯ «PYTHON» У ЗАКЛАДАХ ОСВІТИ ПІД ЧАС ДИСТАНЦІЙНОГО НАВЧАННЯ // “Вересень.” 2023. Vol. 1, № 96.

9. Osadcha K.P., Khromyshev O.V. Розв’язання математичних задач засобами мови програмування Python // Ukrainian Journal of Educational Studies and Information Technology. 2017. Vol. 5, № 1.

10. Дмитренко Т. Методика обробки аудіо-сигналів за допомогою алгоритмів на базі мови програмування Python. // COMPUTER-INTEGRATED TECHNOLOGIES: EDUCATION, SCIENCE, PRODUCTION. 2020. № 41.

11. Farzat F.D.A., Barros M.D.O., Travassos G.H. Evolving JavaScript Code to Reduce Load Time // IEEE Transactions on Software Engineering. 2021. Vol. 47, № 8.

12. Wang Z. et al. An empirical study on bugs in JavaScript engines // Inf Softw Technol. 2023. Vol. 155.

13. Sholikhan M. HTML, CSS dan Javascript // Penerbit Yayasan Prima Agus Teknik. 2022.

14. Xiao X. et al. “Computing” Requirements for Open Source Software: A Distributed Cognitive Approach // J Assoc Inf Syst. 2018. Vol. 19, № 12.

15. Schiappa R. et al. RUBY: Natural Language Processing of French Electronic Medical Records for Breast Cancer Research // JCO Clin Cancer Inform. 2022. № 6.

16. Astrauskas V. et al. How do programmers use unsafe rust? // Proceedings of the ACM on Programming Languages. 2020. Vol. 4, № OOPSLA.

17. Yuan T., Lu J., Li L. An empirical study on Go concurrency bug detection // Gaojishu Tongxin/Chinese High Technology Letters. 2023. Vol. 33, № 4.

18. Comparison of Bubble and Insertion Sort in Rust and Python Language // International Journal of Advanced Trends in Computer Science and Engineering. 2021. Vol. 10, № 2.

19. de Azevedo Y.C.A. et al. SCHOOL ATTENDANCE MANAGEMENT SYSTEM FOR STUDENTS THROUGH FACIAL RECOGNITION // Journal of Engineering and Technology for Industrial Applications. 2022. Vol. 8, № 38.

20. Ogala J., Ogala B., Onyarin J. Comparative Analysis of C, C++, C# and JAVA Programming Languages // Global Scientific Journals. 2020. Vol. 8, № 5.

21. Wirfs-Brock A., Eich B. JavaScript: The first 20 years // Proceedings of the ACM on Programming Languages. 2020. Vol. 4, № HOPL.

22. Jordana A. What Is JavaScript? A Basic Introduction to JS for Beginners // Hostinger Tutorials. 2023.

23. Bugden W., Alahmar A. Rust: The Programming Language for Safety and Performance. 2022.

24. Lattuada A. et al. Verus: Verifying Rust Programs using Linear Ghost Types // Proceedings of the ACM on Programming Languages. 2023. Vol. 7, № OOPSLA1.

ДОДАТОК А

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Продуктивність мов програмування</title>
    <link rel="stylesheet" href="css/style.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <header>
        <h1>Тести продуктивності мов програмування</h1>
        <nav>
            <a href="index.html">Тести продуктивності</a>
            <a href="usability.html">Тести зручності</a>
        </nav>
    </header>
    <main>
        <canvas id="resultsChart" width="800" height="400"></canvas>
    </main>
    <footer>
        <p>Розробник Бараненко Олександр</p>
    </footer>
    <script src="js/script.js"></script>
</body>
</html>
```

Usability.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Usability Survey</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <header>
        <h1>Дослідження зручності для мов програмування</h1>
        <nav>
            <a href="index.html">Тести продуктивності</a>
            <a href="usability.html">Тести зручності</a>
        </nav>
    </header>
    <main>
        <main>
            <h2>Будь ласка, оцініть зручність використання наступних мов
програмування:</h2>
            <p>
```

```
            Використовуйте повзунки нижче, щоб оцінити кожну мову. Оцінки
варіюються від 1 (погано) до 5 (відмінно).
            </p>
        <form id="usabilityForm">
            <!-- Python -->
            <div class="language-survey">
                <h3>Python</h3>
                <div class="question">
                    <label  for="pythonSyntaxRating">Наскільки  ви  задоволені
синтаксисом Python?</label>
                    <input        type="range"        min="1"        max="5"
id="pythonSyntaxRating" name="pythonSyntaxRating" value="1">
                </div>
                <div class="question">
                    <label    for="pythonReadabilityRating">Наскільки    легко
читати код на Python?</label>
                    <input        type="range"        min="1"        max="5"
id="pythonReadabilityRating" name="pythonReadabilityRating" value="1">
                </div>
                <div class="question">
                    <label for="pythonSpeedRating">Як швидко ви можете досягти
результатів за допомогою Python?</label>
                    <input type="range" min="1" max="5" id="pythonSpeedRating"
name="pythonSpeedRating" value="1">
                </div>
            </div>
            <!-- Ruby -->
            <div class="language-survey">
                <h3>Ruby</h3>
                <div class="question">
                    <label   for="rubySyntaxRating">Наскільки  ви  задоволені
синтаксисом Ruby?</label>
                    <input type="range" min="1" max="5" id="rubySyntaxRating"
name="rubySyntaxRating" value="1">
                </div>
                <div class="question">
                    <label for="rubyReadabilityRating">Наскільки легко читати
код на Ruby?</label>
                    <input        type="range"        min="1"        max="5"
id="rubyReadabilityRating" name="rubyReadabilityRating" value="1">
                </div>
                <div class="question">
                    <label for="rubySpeedRating">Як швидко ви можете досягти
результатів за допомогою Ruby?</label>
                    <input  type="range"  min="1"  max="5"  id="rubySpeedRating"
name="rubySpeedRating" value="1">
                </div>
            </div>
            <!-- Go -->
            <div class="language-survey">
                <h3>Go</h3>
```

```
                <div class="question">
                    <label   for="goSyntaxRating">Наскільки   ви   задоволені
синтаксисом мови Go?</label>
                    <input  type="range"  min="1"  max="5"  id="goSyntaxRating"
name="goSyntaxRating" value="1">
                </div>
                <div class="question">
                    <label  for="goReadabilityRating">Наскільки  легко  читати
код мовою Go?</label>
                    <input        type="range"        min="1"        max="5"
id="goReadabilityRating" name="goReadabilityRating" value="1">
                </div>
                <div class="question">
                    <label  for="goSpeedRating">Як  швидко  ви  можете  досягти
результатів за допомогою Go?</label>
                    <input  type="range"  min="1"  max="5"  id="goSpeedRating"
name="goSpeedRating" value="1">
                </div>
            </div>
            <!-- Rust -->
            <div class="language-survey">
                <h3>Rust</h3>
                <div class="question">
                    <label   for="rustSyntaxRating">Наскільки   ви   задоволені
синтаксисом Rust?</label>
                    <input type="range" min="1" max="5" id="rustSyntaxRating"
name="rustSyntaxRating" value="1">
                </div>
                <div class="question">
                    <label for="rustReadabilityRating">Наскільки легко читати
код Rust?</label>
                    <input        type="range"        min="1"        max="5"
id="rustReadabilityRating" name="rustReadabilityRating" value="1">
                </div>
                <div class="question">
                    <label for="rustSpeedRating">Як швидко ви можете досягти
результатів за допомогою Rust?</label>
                    <input  type="range"  min="1"  max="5"  id="rustSpeedRating"
name="rustSpeedRating" value="1">
                </div>
            </div>
            <!-- JavaScript -->
            <div class="language-survey">
                <h3>JavaScript</h3>
                <div class="question">
                    <label   for="jsSyntaxRating">Наскільки   ви   задоволені
синтаксисом JavaScript?</label>
                    <input  type="range"  min="1"  max="5"  id="jsSyntaxRating"
name="jsSyntaxRating" value="1">
                </div>
                <div class="question">
```

```
                    <label for="jsReadabilityRating">Наскільки легко читати
код JavaScript?</label>
                    <input type="range" min="1" max="5"
id="jsReadabilityRating" name="jsReadabilityRating" value="1">
                </div>
                <div class="question">
                    <label for="jsSpeedRating">Як швидко ви можете досягти
результатів за допомогою JavaScript?</label>
                    <input type="range" min="1" max="5" id="jsSpeedRating"
name="jsSpeedRating" value="1">
                </div>
            </div>
            <!-- C++ -->
            <div class="language-survey">
                <h3>C++</h3>
                <div class="question">
                    <label for="cppSyntaxRating">Наскільки ви задоволені
синтаксисом C++?</label>
                    <input type="range" min="1" max="5" id="cppSyntaxRating"
name="cppSyntaxRating" value="1">
                </div>
                <div class="question">
                    <label for="cppReadabilityRating">Наскільки легко читати
код C++?</label>
                    <input type="range" min="1" max="5"
id="cppReadabilityRating" name="cppReadabilityRating" value="1">
                </div>
                <div class="question">
                    <label for="cppSpeedRating">Як швидко ви можете досягти
результатів за допомогою C++?</label>
                    <input type="range" min="1" max="5" id="cppSpeedRating"
name="cppSpeedRating" value="1">
                </div>
            </div>
            <button type="button" id="submitSurvey">Надіслати</button>
        </form>
        <div id="surveyResults" class="hidden">
            <h2>Дякуємо за ваш відгук!</h2>
            <h3>Результати Опитування</h3>
            <table id="resultsTable">
                <thead>
                    <tr>
                        <th>Мова</th>
                        <th>Синтаксис</th>
                        <th>Читабельність</th>
                        <th>Швидкість</th>
                    </tr>
                </thead>
                <tbody>
                </tbody>
            </table>
```

```
        </div>
    </main>
    <footer>
        <p>Розробник Бараненко Олександр</p>
    </footer>
    <script src="js/survey.js"></script>
</body>
</html>
```

Script.js

```
const ctx = document.getElementById('resultsChart').getContext('2d');
const savedResults = [
    { language: "C++", performance1: 10.99, performance2: 0.2314, performance3:
1.295 },
    { language: "Rust", performance1: 46.35, performance2: 0.5374,
performance3: 3.539 },
    { language: "Go", performance1: 9.88, performance2: 0.0834, performance3:
4.652 },
    { language: "Python", performance1: 82.28, performance2: 1.4587,
performance3: 5.565 },
    { language: "JavaScript", performance1: 14.85, performance2: 0.7628,
performance3: 11.802 },
    { language: "Ruby", performance1: 54.28, performance2: 2.6942,
performance3: 122.935 }
];
const labels = savedResults.map(result => result.language);
const performance1 = savedResults.map(result => result.performance1);
const performance2 = savedResults.map(result => result.performance2);
const performance3 = savedResults.map(result => result.performance3);
new Chart(ctx, {
    type: 'bar',
    data: {
        labels,
        datasets: [
            {
                label: 'Фібоначчі',
                data: performance1,
                backgroundColor: '#4caf50',
                borderColor: '#388e3c',
                borderWidth: 1
            },
            {
                label: 'Сортування',
                data: performance2,
                backgroundColor: '#ff9800',
                borderColor: '#e65100',
                borderWidth: 1
            },
            {
                label: 'Матриця',
                data: performance3,
```

```
                backgroundColor: '#2196f3',
                borderColor: '#1e88e5',
                borderWidth: 1
            }
        ]
    },
    options: {
        responsive: true,
        scales: {
            y: {
                beginAtZero: true,
                stacked: false
            },
            x: {
                stacked: false
            }
        },
        plugins: {
            legend: {
                position: 'top'
            }
        }
    }
});

Result.js
document.addEventListener("DOMContentLoaded", function() {
    const                        surveyResultsContainer                        =
document.getElementById("surveyResultsContainer");
    const surveyResults = [
        { language: "Python", syntax: 4, readability: 5, speed: 4 },
        { language: "Ruby", syntax: 3, readability: 4, speed: 3 },
        { language: "Go", syntax: 5, readability: 5, speed: 5 },
        { language: "Rust", syntax: 5, readability: 5, speed: 4 },
        { language: "JavaScript", syntax: 4, readability: 4, speed: 4 },
        { language: "C++", syntax: 3, readability: 3, speed: 5 }
    ];
    surveyResults.forEach(result => {
        const resultElement = document.createElement('div');
        resultElement.classList.add('survey-result');
        resultElement.innerHTML = `
            <h3>${result.language}</h3>
            <p><strong>Syntax Satisfaction:</strong> ${result.syntax}</p>
            <p><strong>Readability:</strong> ${result.readability}</p>
            <p><strong>Speed:</strong> ${result.speed}</p>
        `;
        surveyResultsContainer.appendChild(resultElement);
    });
});
function getQueryParams() {
    const params = new URLSearchParams(window.location.search);
```

```
    return {
        language: params.get('language'),
        syntax: params.get('syntax'),
        readability: params.get('readability'),
        speed: params.get('speed')
    };
}
window.onload = function() {
    const params = getQueryParams();
    document.getElementById('resultLanguage').textContent = params.language;
    document.getElementById('resultSyntax').textContent = params.syntax;
    document.getElementById('resultReadability').textContent                =
params.readability;
    document.getElementById('resultSpeed').textContent = params.speed;
    const performanceScores = {
        "Python": 80,
        "Ruby": 60,
        "Go": 95,
        "Rust": 100,
        "JavaScript": 85,
        "C++": 90
    };
    const performanceScore = performanceScores[params.language] || 0;
    document.getElementById('performanceScore').textContent   =   `Performance
Score: ${performanceScore}`;
};

Survey.js
document.getElementById('submitSurvey').addEventListener('click',  function  ()
{
    const surveyResults = {
        python: {
            syntax: document.getElementById('pythonSyntaxRating').value,
            readability:
document.getElementById('pythonReadabilityRating').value,
            speed: document.getElementById('pythonSpeedRating').value
        },
        ruby: {
            syntax: document.getElementById('rubySyntaxRating').value,
            readability:
document.getElementById('rubyReadabilityRating').value,
            speed: document.getElementById('rubySpeedRating').value
        },
        go: {
            syntax: document.getElementById('goSyntaxRating').value,
            readability: document.getElementById('goReadabilityRating').value,
            speed: document.getElementById('goSpeedRating').value
        },
        rust: {
            syntax: document.getElementById('rustSyntaxRating').value,
```

```
            readability:
document.getElementById('rustReadabilityRating').value,
            speed: document.getElementById('rustSpeedRating').value
        },
        js: {
            syntax: document.getElementById('jsSyntaxRating').value,
            readability: document.getElementById('jsReadabilityRating').value,
            speed: document.getElementById('jsSpeedRating').value
        },
        cpp: {
            syntax: document.getElementById('cppSyntaxRating').value,
            readability:
document.getElementById('cppReadabilityRating').value,
            speed: document.getElementById('cppSpeedRating').value
        }
    };
    for (const language in surveyResults) {
        for (const attribute in surveyResults[language]) {
            if (surveyResults[language][attribute] === "") {
                alert(`Будь    ласка,    оцініть    всі    параметри    для
${language.toUpperCase()}!`);
                return;
            }
        }
    }
    const                                tableBody                                =
document.getElementById('resultsTable').getElementsByTagName('tbody')[0];
    tableBody.innerHTML = '';
    for (const language in surveyResults) {
        const row = document.createElement('tr');
        const languageCell = document.createElement('td');
        let    displayLanguage    =    language.charAt(0).toUpperCase()    +
language.slice(1);
        if (language === 'js') displayLanguage = 'JavaScript';
        if (language === 'cpp') displayLanguage = 'C++';
        languageCell.textContent = displayLanguage;
        row.appendChild(languageCell);
        const syntaxCell = document.createElement('td');
        syntaxCell.textContent = surveyResults[language].syntax;
        row.appendChild(syntaxCell);
        const readabilityCell = document.createElement('td');
        readabilityCell.textContent = surveyResults[language].readability;
        row.appendChild(readabilityCell);
        const speedCell = document.createElement('td');
        speedCell.textContent = surveyResults[language].speed;
        row.appendChild(speedCell);
        tableBody.appendChild(row);
    }
    document.getElementById('surveyResults').classList.remove('hidden');

});
```

Style.css

```
body {
    font-family: 'Arial', sans-serif;
    margin: 0;
    padding: 0;
    background: #f3f4f7;
    color: #333;
    line-height: 1.6;
    box-sizing: border-box;
}
header {
    background: #4caf50;
    color: white;
    padding: 20px 10px;
    text-align: center;
    box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
    border-bottom: 4px solid #388e3c;
}
header h1 {
    margin: 0;
    font-size: 28px;
    font-weight: 600;
    letter-spacing: 1.2px;
    text-transform: uppercase;
}
nav {
    margin-top: 10px;
}
nav a {
    color: white;
    text-decoration: none;
    margin: 0 15px;
    font-size: 16px;
    letter-spacing: 1.2px;
    text-transform: uppercase;
    transition: color 0.3s ease;
}
nav a:hover {
    color: #e8f5e9;
    text-decoration: underline;
}
main {
    margin: 30px auto;
    padding: 30px;
    max-width: 900px;
    background: white;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.1);
    border-radius: 12px;
    transition: transform 0.3s ease-in-out, box-shadow 0.3s ease, margin 0.3s
ease;
}
```

```
main:hover {
    box-shadow: 0 6px 20px rgba(0, 0, 0, 0.2);
}
form h3 {
    background: #4caf50;
    color: white;
    padding: 15px;
    margin: 20px 0;
    border-radius: 8px;
    font-size: 20px;
    text-align: center;
    letter-spacing: 1px;
}
.question {
    margin: 20px 0;
    display: flex;
    flex-direction: column;
    justify-content: center;
}
label {
    font-size: 16px;
    margin-bottom: 8px;
    color: #555;
}
input[type="range"] {
    width: 100%;
    appearance: none;
    height: 12px;
    background: #ddd;
    border-radius: 5px;
    outline: none;
    cursor: pointer;
    transition: background 0.3s ease-in-out;
}
input[type="range"]:hover {
    background: #bbb;
}
input[type="range"]::-webkit-slider-thumb {
    appearance: none;
    width: 24px;
    height: 24px;
    background: #4caf50;
    border-radius: 50%;
    cursor: pointer;
    transition: background 0.3s ease;
}
input[type="range"]::-moz-range-thumb {
    width: 24px;
    height: 24px;
    background: #4caf50;
    border-radius: 50%;
```

```
    cursor: pointer;
    transition: background 0.3s ease;
}
input[type="range"]:hover::-webkit-slider-thumb,
input[type="range"]:hover::-moz-range-thumb {
    background: #45a049;
}
button {
    background-color: #4caf50;
    color: white;
    padding: 16px 28px;
    border: none;
    cursor: pointer;
    font-size: 18px;
    border-radius: 8px;
    width: 100%;
    transition: background-color 0.3s ease, transform 0.2s ease;
}
button:hover {
    background-color: #45a049;
    transform: translateY(-5px) scale(1.05); /* Кнопка "піднімається" при
наведенні */
}
#surveyResults {
    text-align: center;
    padding: 25px;
    background: #e8f5e9;
    border-radius: 10px;
    margin-top: 30px;
    box-shadow: 0 4px 15px rgba(0, 0, 0, 0.1);
    transition: box-shadow 0.3s ease, transform 0.3s ease;
}
#surveyResults:hover {
    box-shadow: 0 6px 25px rgba(0, 0, 0, 0.15);
    transform: translateY(-5px);  /* Легке піднімання при наведенні */
}
#resultsTable {
    width: 100%;
    margin: 20px 0;
    border-collapse: collapse;
    text-align: center;
}
#resultsTable th, #resultsTable td {
    padding: 16px;
    border: 1px solid #ddd;
    font-size: 16px;
    color: #555;
    transition: background-color 0.3s ease;
}
#resultsTable th {
    background: #4caf50;
```

```
    color: white;
    font-weight: bold;
}
#resultsTable tr:nth-child(even) {
    background: #f9f9f9;
}
#resultsTable tr:hover {
    background-color: #f1f8f1;
    transition: background-color 0.3s ease;
}
#resultsTable td {
    font-weight: 600;
}
```

ДОДАТОК В

Тести фібоначчі

C++

```
#include <iostream>
#include <chrono>
using namespace std;
int fibonacci(int n) {
    if (n <= 1) {
        return n;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
int main() {
    int n = 35;
    auto start = chrono::high_resolution_clock::now();
    int result = fibonacci(n);
    auto end = chrono::high_resolution_clock::now();
    chrono::duration<double> duration = end - start;
    cout << "Число Фібоначчі для n=" << n << ": " << result << endl;
    cout << "Час виконання: " << duration.count() << " секунд" << endl;
    return 0;
}
```

Rust

```
use std::time::Instant;
fn fibonacci(n: u32) -> u32 {
    if n <= 1 {
        n
    } else {
        fibonacci(n - 1) + fibonacci(n - 2)
    }
}
fn main() {
    let n = 35;
    let start = Instant::now();
    let result = fibonacci(n);
    let duration = start.elapsed();
    println!("Число Фібоначчі для n={}: {}", n, result);
    println!("Час виконання: {:?}", duration);
}
```

Go

```
package main
import (
"fmt"
"time"
)
func fibonacci(n int) int {
if n <= 1 {
return n
```

```
} else {
return fibonacci(n-1) + fibonacci(n-2)
}
}
func main() {
n := 35
start := time.Now()
result := fibonacci(n)
elapsed := time.Since(start)
fmt.Printf("Число Фібоначчі для n=%d: %d\n", n, result)
fmt.Printf("Час виконання: %s\n", elapsed)
}
```

```
Python
import time
def fibonacci(n):
    if n <= 1:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)
n = 35
start_time = time.time()
result = fibonacci(n)
end_time = time.time()
print(f"Число Фібоначчі для n={n}: {result}")
print(f"Час виконання: {end_time - start_time} секунд")
```

```
JavaScript
function fibonacci(n) {
    if (n <= 1) {
        return n;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
let n = 35;
console.time('fibonacci');
let result = fibonacci(n);
console.timeEnd('fibonacci');
console.log(`Число Фібоначчі для n=${n}: ${result}`);
```

```
Ruby
def fibonacci(n)
  if n <= 1
    return n
  else
    return fibonacci(n - 1) + fibonacci(n - 2)
  end
end
n = 35
start_time = Time.now
```

```
result = fibonacci(n)
end_time = Time.now
puts "Число Фібоначчі для n=#{n}: #{result}"
puts "Час виконання: #{end_time - start_time} секунд"
```

Сортування

C++

```
#include <iostream>
#include <vector>
#include <chrono>
using namespace std;
vector<int> quick_sort(vector<int>& arr) {
    if (arr.size() <= 1) return arr;
    int pivot = arr[arr.size() / 2];
    vector<int> left, middle, right;
    for (int num : arr) {
        if (num < pivot)
            left.push_back(num);
        else if (num == pivot)
            middle.push_back(num);
        else
            right.push_back(num);
    }
    left = quick_sort(left);
    right = quick_sort(right);
    left.insert(left.end(), middle.begin(), middle.end());
    left.insert(left.end(), right.begin(), right.end());
    return left;
}
int main() {
    vector<int> arr(100000);
    for (int i = 0; i < 100000; ++i) {
        arr[i] = 100000 - i;
    }
    auto start = chrono::high_resolution_clock::now();
    vector<int> sorted_arr = quick_sort(arr);
    auto end = chrono::high_resolution_clock::now();
    chrono::duration<double> duration = end - start;
    cout << "Час виконання: " << duration.count() << " секунд" << endl;
    return 0;
}
```

Rust

```
use std::time::Instant;
fn quick_sort(arr: &mut Vec<i32>) {
    if arr.len() <= 1 {
        return;
    }
    let pivot = arr[arr.len() / 2];
    let mut left = Vec::new();
    let mut middle = Vec::new();
```

```
    let mut right = Vec::new();
    for &num in arr.iter() {
        if num < pivot {
            left.push(num);
        } else if num == pivot {
            middle.push(num);
        } else {
            right.push(num);
        }
    }
    quick_sort(&mut left);
    quick_sort(&mut right);
    arr.clear();
    arr.append(&mut left);
    arr.append(&mut middle);
    arr.append(&mut right);
}
fn main() {
    let mut arr: Vec<i32> = (1..=100000).rev().collect();
    let start = Instant::now();
    quick_sort(&mut arr);
    let duration = start.elapsed();
    println!("Час виконання: {:?}", duration);
}
```

```
Go
package main
import (
"fmt"
"time"
)
func quickSort(arr []int) []int {
if len(arr) <= 1 {
return arr
}
pivot := arr[len(arr)/2]
var left, middle, right []int
for _, num := range arr {
if num < pivot {
left = append(left, num)
} else if num == pivot {
middle = append(middle, num)
} else {
right = append(right, num)
}
}
left = quickSort(left)
right = quickSort(right)
return append(append(left, middle...), right...)
}
```

```
func main() {
arr := make([]int, 100000)
for i := 0; i < 100000; i++ {
arr[i] = 100000 - i
}
start := time.Now()
sortedArr := quickSort(arr)
elapsed := time.Since(start)
fmt.Printf("Час виконання: %s\n", elapsed)
}
```

```
Python
import time
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return quick_sort(left) + middle + quick_sort(right)
arr = [i for i in range(100000, 0, -1)]
start_time = time.time()
sorted_arr = quick_sort(arr)
end_time = time.time()
print(f"Час виконання: {end_time - start_time} секунд")
```

```
JavaScript
function quickSort(arr) {
    if (arr.length <= 1) {
        return arr;
    }
    const pivot = arr[Math.floor(arr.length / 2)];
    const left = arr.filter(x => x < pivot);
    const middle = arr.filter(x => x === pivot);
    const right = arr.filter(x => x > pivot);
    return [...quickSort(left), ...middle, ...quickSort(right)];
}
let arr = Array.from({ length: 100000 }, (_, i) => 100000 - i);
console.time('quickSort');
let sortedArr = quickSort(arr);
console.timeEnd('quickSort');
```

```
Ruby
def quick_sort(arr)
  return arr if arr.length <= 1
  pivot = arr[arr.length / 2]
  left = arr.select { |x| x < pivot }
  middle = arr.select { |x| x == pivot }
  right = arr.select { |x| x > pivot }
  quick_sort(left) + middle + quick_sort(right)
```

```
end
arr = (1..100000).to_a.reverse
start_time = Time.now
sorted_arr = quick_sort(arr)
end_time = Time.now
puts "Час виконання: #{end_time - start_time} секунд"
```

Додавання та множення матриць

C++

```
#include <vector>
#include <chrono>
#include <cstdlib>
using namespace std;
vector<vector<int>>   add_matrices(const   vector<vector<int>>&   a,   const
vector<vector<int>>& b) {
    int rows = a.size();
    int cols = a[0].size();
    vector<vector<int>> result(rows, vector<int>(cols, 0));
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            result[i][j] = a[i][j] + b[i][j];
        }
    }
    return result;
}
vector<vector<int>>  multiply_matrices(const  vector<vector<int>>&  a,  const
vector<vector<int>>& b) {
    int rows_a = a.size();
    int cols_a = a[0].size();
    int rows_b = b.size();
    int cols_b = b[0].size();
    if (cols_a != rows_b) {
        throw runtime_error("Матриці не сумісні для множення");
    }
    vector<vector<int>> result(rows_a, vector<int>(cols_b, 0));
    for (int i = 0; i < rows_a; ++i) {
        for (int j = 0; j < cols_b; ++j) {
            for (int k = 0; k < cols_a; ++k) {
                result[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    return result;
}
int main() {
    int n = 500;
    vector<vector<int>> matrix_a(n, vector<int>(n, 1));
    vector<vector<int>> matrix_b(n, vector<int>(n, 2));
    auto start = chrono::high_resolution_clock::now();
    auto result_add = add_matrices(matrix_a, matrix_b);
```

```
    auto end = chrono::high_resolution_clock::now();
    chrono::duration<double> duration_add = end - start;
    cout << "Час виконання додавання матриць: " << duration_add.count() << "
секунд" << endl;
    start = chrono::high_resolution_clock::now();
    auto result_mul = multiply_matrices(matrix_a, matrix_b);
    end = chrono::high_resolution_clock::now();
    chrono::duration<double> duration_mul = end - start;
    cout << "Час виконання множення матриць: " << duration_mul.count() << "
секунд" << endl;
    double total_duration = duration_add.count() + duration_mul.count();
    cout << "Загальний час виконання обох операцій: " << total_duration << "
секунд" << endl;
    return 0;
}
```

Rust

```
use std::time::Instant;
fn add_matrices(a: &Vec<Vec<i32>>, b: &Vec<Vec<i32>>) -> Vec<Vec<i32>> {
    let rows = a.len();
    let cols = a[0].len();
    let mut result = vec![vec![0; cols]; rows];
    for i in 0..rows {
        for j in 0..cols {
            result[i][j] = a[i][j] + b[i][j];
        }
    }
    result
}
fn multiply_matrices(a: &Vec<Vec<i32>>, b: &Vec<Vec<i32>>) -> Vec<Vec<i32>> {
    let rows_a = a.len();
    let cols_a = a[0].len();
    let rows_b = b.len();
    let cols_b = b[0].len();
    if cols_a != rows_b {
        panic!("Матриці не сумісні для множення");
    }
    let mut result = vec![vec![0; cols_b]; rows_a];
    for i in 0..rows_a {
        for j in 0..cols_b {
            for k in 0..cols_a {
                result[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    result
}
fn main() {
    let n = 700;
    let matrix_a = vec![vec![1; n]; n];
    let matrix_b = vec![vec![2; n]; n];
```

```
    let start = Instant::now();
    let result_add = add_matrices(&matrix_a, &matrix_b);
    let duration_add = start.elapsed();
    println!("Час виконання додавання матриць: {:?}", duration_add);
    let start = Instant::now();
    let result_mul = multiply_matrices(&matrix_a, &matrix_b);
    let duration_mul = start.elapsed();
    println!("Час виконання множення матриць: {:?}", duration_mul);
    let total_duration = duration_add + duration_mul;
    println!("Спільний час виконання обох операцій: {:?}", total_duration);
}
```

```
Go
package main
import (
"fmt"
"sync"
"time"
)
func addMatrices(a, b [][]int, result [][]int, wg *sync.WaitGroup) {
defer wg.Done()
rows := len(a)
cols := len(a[0])
for i := 0; i < rows; i++ {
for j := 0; j < cols; j++ {
result[i][j] = a[i][j] + b[i][j]
}
}
}
func multiplyMatrices(a, b [][]int, result [][]int, wg *sync.WaitGroup) {
defer wg.Done()
rows := len(a)
cols := len(b[0])
common := len(b)
for i := 0; i < rows; i++ {
for j := 0; j < cols; j++ {
for k := 0; k < common; k++ {
result[i][j] += a[i][k] * b[k][j]
}
}
}
}

func main() {
n := 500
matrixA := make([][]int, n)
matrixB := make([][]int, n)
for i := 0; i < n; i++ {
matrixA[i] = make([]int, n)
matrixB[i] = make([]int, n)
for j := 0; j < n; j++ {
```

```
matrixA[i][j] = 1
matrixB[i][j] = 2
}
}
resultAdd := make([][]int, n)
resultMul := make([][]int, n)
for i := 0; i < n; i++ {
resultAdd[i] = make([]int, n)
resultMul[i] = make([]int, n)
}
var totalDuration time.Duration
start := time.Now()
var wg sync.WaitGroup
wg.Add(1)
go addMatrices(matrixA, matrixB, resultAdd, &wg)
wg.Wait()
durationAdd := time.Since(start)
totalDuration += durationAdd
fmt.Printf("Час виконання додавання матриць: %s\n", durationAdd)
start = time.Now()
wg.Add(1)
go multiplyMatrices(matrixA, matrixB, resultMul, &wg)
wg.Wait()
durationMul := time.Since(start)
totalDuration += durationMul
fmt.Printf("Час виконання множення матриць: %s\n", durationMul)
fmt.Printf("Спільний час виконання обох операцій: %s\n", totalDuration)
}
```

```
Pyhon
import time
import numpy as np
matrix_a = np.random.randint(1, 10, size=(500, 500))
matrix_b = np.random.randint(1, 10, size=(500, 500))
total_time = 0
start_time = time.time()
result_add = np.add(matrix_a, matrix_b)
end_time = time.time()
add_time = end_time - start_time
total_time += add_time
print(f"Час виконання додавання матриць: {add_time:.6f} секунд")
start_time = time.time()
result_mul = np.matmul(matrix_a, matrix_b)
end_time = time.time()
mul_time = end_time - start_time
total_time += mul_time
print(f"Час виконання множення матриць: {mul_time:.6f} секунд")
print(f"Загальний час виконання обох операцій: {total_time:.6f} секунд")
```

```
JavaScript
function addMatrices(a, b) {
```

```
    const rows = a.length;
    const cols = a[0].length;
    let result = Array.from({ length: rows }, () => Array(cols).fill(0));
    for (let i = 0; i < rows; i++) {
        for (let j = 0; j < cols; j++) {
            result[i][j] = a[i][j] + b[i][j];
        }
    }
    return result;
}
function multiplyMatrices(a, b) {
    const rows = a.length;
    const cols = b[0].length;
    const common = b.length;
    let result = Array.from({ length: rows }, () => Array(cols).fill(0));
    for (let i = 0; i < rows; i++) {
        for (let j = 0; j < cols; j++) {
            for (let k = 0; k < common; k++) {
                result[i][j] += a[i][k] * b[k][j];
            }
        }
    }
    return result;
}
let n = 500; // Змінити для ускладнення
let matrix_a = Array.from({ length: n }, () => Array(n).fill(1));
let matrix_b = Array.from({ length: n }, () => Array(n).fill(2));
let startAdd = performance.now();
let resultAdd = addMatrices(matrix_a, matrix_b);
let endAdd = performance.now();
let timeAdd = endAdd - startAdd;
console.log(`Час виконання додавання матриць: ${(timeAdd / 1000).toFixed(6)}
секунд`);
let startMul = performance.now();
let resultMul = multiplyMatrices(matrix_a, matrix_b);
let endMul = performance.now();
let timeMul = endMul - startMul;
console.log(`Час виконання множення матриць: ${(timeMul / 1000).toFixed(6)}
секунд`);
let totalTime = timeAdd + timeMul;
console.log(`Загальний час виконання обох операцій: ${(totalTime /
1000).toFixed(6)} секунд`);
```

Ruby

```
def add_matrices(a, b)
rows = a.length
cols = a[0].length
result = Array.new(rows) { Array.new(cols, 0) }
rows.times do |i|
cols.times do |j|
result[i][j] = a[i][j] + b[i][j]
```

```
end
end
result
end
def multiply_matrices(a, b)
rows_a = a.length
cols_a = a[0].length
rows_b = b.length
cols_b = b[0].length
if cols_a != rows_b
raise "Матриці не сумісні для множення"
end
result = Array.new(rows_a) { Array.new(cols_b, 0) }
rows_a.times do |i|
cols_b.times do |j|
cols_a.times do |k|
result[i][j] += a[i][k] * b[k][j]
end
end
end
result
n = 500
matrix_a = Array.new(n) { Array.new(n, 1) }
matrix_b = Array.new(n) { Array.new(n, 2) }
start_time = Time.now
result_add = add_matrices(matrix_a, matrix_b)
end_time = Time.now
puts "Час виконання додавання матриць: #{end_time - start_time} секунд"
start_time = Time.now
result_mul = multiply_matrices(matrix_a, matrix_b)
end_time = Time.now
puts "Час виконання множення матриць: #{end_time - start_time} секунд"
```