5. VMware: the new Redis home [Електронний ресурс]. – Режим доступу: http://antirez.com/post/vmware-the-new-redis-home.html.

Анотація. У статті описані основні засоби, які були використані при розробці інформаційно-аналітичної системи «SmartCity», визначені переваги використання конструктора програмного інтерфейсу для взаємодії з нереляційних базою даних.

Ключові слова: конструктор програмного інтерфейсу, API, нереляційна база даних, Redis.

УДК 004.041

ПОРІВНЯННЯ КРОС-ПЛАТФОРМНИХ МОВ ПРОГРАМУВАННЯ JAVA ТА QT

Ройко Є.
Мелітопольський державний педагогічний університет імені Богдана Хмельницького, м. Мелітополь
e-mail: Soldatko@mail.ru
Сіциліцин Ю.
Таврійський державний агротехнологічний університет, м. Мелітополь
e-mail: yurarud@mail.ru

При виборі засобів для розробки великого програмного проекту необхідно врахувати безліч різних аспектів, найбільш важливим з яких є мова програмування, тому що він в значній мірі визначає інші доступні засоби. Наприклад, для розробки графічного інтерфейсу розробникам необхідна GUI-бібліотека, що надає готові елементи інтерфейсу, такі, як кнопки і меню. Так як вибір GUI-бібліотеки дуже впливає на розробку проекту, часто її вибір здійснюється першим, а мова програмування визначається з числа доступних для цієї бібліотеки мов. Зазвичай, мова програмування визначається бібліотекою однозначно.

Метою цієї статті є порівняння C ++ / Qt і Java / AWT / Swing.

Відмінною особливістю Java в порівнянні з іншими мовами програмування загального призначення є забезпечення високої продуктивності програмування, ніж продуктивність роботи програми або ефективність використання ним пам’яті.

Однак проведене дослідження показує, що на практиці збірка "сміття" і інші можливості Java не мають великого впливу на продуктивність програмування. Одна з класичних моделей оцінки програмного забезпечення CoCoMo, запропонована Barry Boehm, зумовлює вартість і терміни розробки програмного продукту на основі вартісних коефіцієнтів, які враховують такі чинники, як сумарний досвід програмування розробника, досвід

програмування на заданому мовою, бажана надійність програми і т.д. Boehm пише, що незалежно від рівня використовуваної мови, початкові трудовитрати завжди високі.

Підводячи підсумок: обидва дослідження і практика спростовують твердження, що Java забезпечує програмістам кращу продуктивність програмування, ніж C ++.

Пояснення того, чому Java-програми повільніше C ++ програм, полягає в наступному. C ++ програми компілюються компілятором C ++ в двійковий формат, який потім виповнюється безпосередньо процесором; таким чином, виконання програми здійснюється апаратними засобами. (Це дещо спрощено, так як більшість сучасних процесорів виконують мікрокод, але це не принципово при обговоренні даного питання.) З іншого боку, компілятор Java компілює вихідний код в "байт-код", який безпосередньо виконується не процесором, а за допомогою іншого програмного забезпечення, віртуальної машини Java (Java Virtual Machine, JVM). У свою чергу, JVM виповнюється процесором. Таким чином, виконання байт-коду Java-програм здійснюється не швидкими апаратними засобами, а за допомогою більш повільної програмної емуляції.

Для підвищення продуктивності роботи Java-програм були розроблені "Just in Time" (JIT) компілятори, але універсального рішення цієї проблеми не існує.

Java і C ++ використовують різні підходи в управлінні пам’яттю. У C ++ управління пам’яттю повністю здійснюється програмістом, тобто в міру необхідності розподіл і звільнення пам’яті повинно виконуватися програмістом. Якщо програміст забуває звільнити раніше отриману пам’ять, виникає "витік пам’яті". Якщо під час роботи програми відбудеться лише одна така витік, проблем не виникне, так як після завершення роботи програми операційна система звільнить всю раніше використану їм пам’ять. Але якщо витоку пам’яті відбуватимуться постійно (наприклад, якщо користувач буде періодично виконувати певні дії), використання пам’яті додатком буде рости аж до повного її витрати з подальшим можливим відмовою системи.

Java забезпечує автоматичне звільнення невикористовуваної пам’яті. Поряд з розподілом пам’яті програмістом JVM веде облік всіх використовуваних блоків пам’яті і покажчиків на них. Якщо блок пам’яті більше не використовується, він може бути звільнений. Це забезпечує процес, який називається "збіркою сміття". Він періодично викликається JVM, перевіряє всі використовувані блоки пам’яті і звільняє ті з них, на які відсутні покажчики.

Прибирання сміття дуже зручна, але за її використання доводиться розплачуватися великим споживанням пам’яті і низькою проізодітельний ... Програмісти C ++ можуть (і повинні) звільняти блоки пам’яті відразу після того, як вони перестали бути потрібні. З Java блоки не звільняються до чергового виклику збирача сміття, періодичність роботи якого залежить від використовуються під реалізації JVM. Prechtelt надає цифрові дані, стверджуючи, що в середньому, (...) і з імовірністю 80% Java-програми використовують на 32 MB (або 297%) пам’яті більше, ніж C / C ++ програми (...). До того ж до великої витрати пам’яті процес збірки сміття вимагає додаткової процесорної потужності, яка в результаті стає недоступною з додатком, і це призводить до уповільнення його роботи. Тому періодична робота збирача сміття може призводити до "заморожування" Java-програми на кілька секунд.

В результаті цього обговорення ми переконалися в тому, що при порівнянній продуктивності програмування C ++ забезпечує додаткам набагато кращі, ніж Java, продуктивність роботи і ефективність використання пам’яті.

Ми порівняли дві платформи: Java / AWT / Swing і C ++ / Qt, оцінивши їх придатність для ефективної розробки високопродуктивних додатків з призначеним для користувача графічним інтерфейсом. У той час, як Java-платформа забезпечує розробникам порівнянну продуктивність програмування, платформа C ++ / Qt забезпечує додаткам кращу продуктивність і ефективність використання пам’яті. Також C ++ виграє за рахунок більш кращих засобів розробки.

Що стосується порівняння GUI-бібліотек, Swing і Qt, то видно, що більш гірша продуктивність Java-програм робить платформу Java / Swing менш придатною для розробки GUI-додатків, навіть при порівнянному досвіді програмування. На відміну від Swing, Qt не нав’язує програмістові парадигму програмування Model-View-Controller, тому в результаті Qt-програми виходять більш короткими..

Література

1. Сравнение Qt и Java - [Електронний ресурс] / - Режим доступу: http://citforum.ck.ua/programming/application/java_qt.shtml.

2. Сравнение Qt и Java - [Електронний ресурс] / - Режим доступу: https://www.linux.org.ru/news/doc/293617

3. Какой язык программирования изучить новичку для написания GUI приложений? - [Електронний ресурс] / - Режим доступу: https://toster.ru/q/237828

Анотація. У статті розробки настільних додатків за допомогою мов програмування Qt та Java. Поставлено проблема про те, як обрати мову програмування та графічний інтерфейс. Проведений аналіз Qt та Java, їх переваг та недоліків. Зроблено висновок про те, яку мову необхідно вибирати у залежності від типу додатку, що розробляється.

Ключові слова. Qt. Java. C++. AWT. Swing.

УДК 517.523-37

ДОСЛІДЖЕННЯ НЕСКІНЧЕННИХ ДОБУТКІВ НА ЗБІЖНІСТЬ ШЛЯХОМ ВИКОРИСТАННЯ ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

Рубцов М., Раділова Х., Сурженко В.
Мелітопольський державний педагогічний університет
імені Богдана Хмельницького,
м. Мелітополь
e-mail: rubtsovnik3077@mail.ru
e-mail: radilova1997@mail.ru
e-mail: missis.surzhenko@mail.ru

Постановка проблеми. Дослідження на збіжність в нескінченних добутках (як і в числових та функціональних рядах) має велике значення для їх використання. Якщо теорія та практична сфера дослідження рядів розроблена досить достатньо, то для нескінченних добутків цього сказати неможливо [1].

Аналіз останніх досліджень і публікацій. Дослідженням нескінченних добутків присвячено багато робіт [1, 4, 5]. В основному матеріал, викладений у навчальній літературі, носить загальний характер, а в численних монографіях з даної тематики розглянуті більш вузькі питання проблеми «Нескінченні добутки». Актуальність цієї роботи обумовлена, з одного боку, великим інтересом до теми «Нескінченні добутки» в сучасній науці, з іншого боку, її недостатньою розробленістю.

Ціллю статті було дослідження нескінченних добутків на збіжність шляхом використання обчислювальної техніки.

Виклад основного матеріалу. Поняття нескінченного добутку дуже близько до поняття числового ряду [3].

Розглянемо нескінченно числову послідовність $u_1, u_2, ..., u_n, ...$

і формально утворимо з елементів цієї послідовності вираз виду:

$$u_1 + u_2 + ... + u_n + ... = \sum_{n=1}^{\infty} u_n . \qquad (1)$$

Вираз (1) прийнято називати числовим рядом або просто рядом.

Нехай дана нескінченна числова послідовність $p_1, p_2, ..., p_n, ...$.

Записаний формально вираз вигляду