АНАЛІЗ СИСТЕМ ПРОГРАМУВАННЯ ДЛЯ НАПИСАННЯ ПРОГРАММ РОЗРАХУНКУ СЄНДВІЧ ПАКЕТІВ

д-р техн. наук, проф. С.С. Добротворський, канд. техн. наук, доц. Є.В. Басова, асп. В.О. Приходько, Національний технічний університет "Харківський політехнічний інститут", м. Харків

Для розрахунку обробки сєндвич пакетів необхідне програмне забезпечення котре дозволить інтегрувати накопиченні розрахункові данні, та практичні данні при обробці сєндвич пакеті (далі ПЗ). Якість ПЗ залежить від отриманих данних та програмістів, і мов програмування, використаних під час розробки .

Метою є проаналізувати корисні мови для написання сторінки для розрахунків в браузері і серед них обрати найбільш оптимальні.

Мови програмування розділимо на кілька груп за різними ознаками:

Низький рівень

Серед характеристик часто зустрічаються: обмеження на абстракції даних, сильна статична Типізація, відсутність проміжного середовища виконання, прямий доступ до пам'яті.

+ Повний контроль практично над усім; ви використовуєте тільки те, що вам потрібно.

- Додатковий контроль означає додаткові складності, які можуть зробити начебто прості завдання важчими в реалізації.

+ Більше контроль над пам'яттю; ви можете зробити те, що практично неможливо в інших мовах.

- Керувати пам'яттю може швидко стати дуже складним.

+ Дозволити вам краще зрозуміти, що відбувається за лаштунками в високорівневих мовах і навчить цінувати абстракції.

- Легко закопатися в синтаксисі і дрібних деталях замість того, щоб розуміти концепцію і загальну картину.

+ Мотивує думати про ефективності.

- Потрібна попередня оптимізація.

+ Мотивує думати про архітектурі наперед.

- Зміни в поганій архітектурі можуть бути болючими. А гарну архітектуру важко придумати.

- Щодо бідна стандартна бібліотека означає, що ви повинні часто покладатися на третіх осіб або винаходити колесо.

- Необхідно часто вставляти допоміжні шматки коду (boilerplate), що збільшує час на розробку.

Середній рівень

Серед характеристик часто зустрічаються: фокус на абстракціях, сильна статична типізація, Виконавча, обмеження на прямий доступ до пам'яті.

Приклади: Java, C #.

+ Керувати пам'яттю необов'язково, але при бажанні ви можете це робити самостійно.

- До сих пір потрібно розуміти, як влаштована пам'ять і як працює прибирання сміття, але мова цього навчання не сприяє.

+ Багаті стандартні бібліотеки.

- Багато абстракції заважають новачкові в освоєнні концепцій, тому що незрозуміло, чому вони створені саме таким чином.

+ Компілює в байт-код, що спрощує взаємодію з іншими мовами.

- Байт-код вимагає встановленої середовища виконання.

- До сих пір потрібно часто вставляти стандартні шматки коду (boilerplate), незважаючи на наявність абстракцій.

Високий рівень

Серед характеристик часто зустрічаються: сильне абстрагування, динамічна або слабка типізація, повністю незалежне управління пам'яттю або наявність середовища виконання.

Приклади: Python, Ruby, JavaScript, Common Lisp.

+ Абстракції роблять складні завдання простими.

- Надбудови для реалізації абстракцій знижують продуктивність.

+ В цілому все просто і інтуїтивно, навіть при внесенні змін.

- Архітектура може страждати, тому що досить просто вносити зміни майже в будь-якому місці замість того, щоб вносити їх там, де потрібно.

+ Порівняно великі стандартні бібліотеки означають, що те, що ви хочете зробити, швидше за все вже реалізовано і доступно.

- Через приховані деталей складно з'ясовувати причини виникнення проблем, коли вони з'являються.

+ Менше вставок стандартного коду – синтаксис значно простіше.

- Динамічна типізація ускладнює пошук помилок без запуску коду.

Виходячи з опису мов програмування зупинимось на JavaSkript тому що:

JavaScript – гнучка мова, яку можна використовувати як для фронтенда, так і для бекенд. Це гарна мова для початківців, оскільки в ньому мало налаштувань, і можна почати писати код прямо в браузері.

У JavaScript велика спільнота, і для його вивчення в Мережі є багато корисних матеріалів.

Значно розширює можливості JS програмна платформа Node.js. З її допомогою код, написаний на JS, можна запускати без браузера на бекенд. А наявність величезної кількості готових рішень в пакетної екосистемі npm дозволяє розробнику не витрачати час на створення більшості рішень.