

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»



Факультет комп'ютерних наук та технологій
Кафедра «Комп'ютерні системи та мережі»

ШКРЯБЕЦЬ ВОЛОДИМИР ІГОРОВИЧ
Група КНТ-622м

СИСТЕМА АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ КОДУ НА
МОВІ ВИСОКОГО РІВНЯ

АВТОРЕФЕРАТ

магістерської роботи на здобуття освітньо-кваліфікаційного
рівня «магістр» 123 «Комп'ютерна інженерія»
освітньої програми «Спеціалізовані комп'ютерні системи»

2023 р.

Магістерська робота є рукопис.

Робота виконана в Національному університеті «Запорізька політехніка», на кафедрі комп'ютерних систем та мереж

Керівник

кандидат технічних наук, доцент
Кудерметов Равіль Камілович,
Національний університет «Запорізька
політехніка», доцент кафедри
комп'ютерних систем та мереж

**Офіційний
рецензент:**

Степаненко Олександр Олексійович,
к.т.н., доцент кафедри програмних засобів
НУ «Запорізька політехніка»

Захист відбудеться "20" грудня 2023 р.

Секретар екзаменаційної комісії, ст. викладач кафедри
комп'ютерних систем та мереж **Н.А. Куликовська**

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність теми. Розробка мультиплатформеного застосунку, тобто під різні операційні системи (ОС), зазвичай означає, що одну і ту ж логіку потрібно буде писати декілька разів для кожної з обраних ОС на притаманній мові програмування. Це призводить до зайвого витрачання людських та фінансових ресурсів.

Існує декілька підходів для оптимізації мультиплатформеної розробки, такі як загальна кодова база (фреймворки), загальні мови програмування (наприклад C++, C) або використання скриптів (наприклад lua/javascript). Але це працює допоки застосунок нескладний та й має свої недоліки. Загалом це неможливість, або складність використання таких підходів в умовах коли проєкт вже в розробці і треба частину перевести на загальні рейки. Доволі часто це призводить до збільшення часу розробки, або ж може вийти непритаманний інтерфейс користувача. Також крім спеціалістів з розробки під конкретну ОС потрібні ще додаткові розробники зі спеціалізованим досвідом.

Системи машинного перекладу з однієї мови високого рівня на іншу далі будемо називати транскompілятором. Транскompіляція може бути використана для створення версій програмного забезпечення для різних мов або платформ. Це може полегшити підтримку і розширення програм у різних середовищах. Застосування транскompіляторів дозволяє об'єднати функціонал різних мов програмування в одному проєкті, що може бути корисним для певних завдань або команд розробників.

Транскompілятори можуть бути використані для перетворення вихідного коду для оптимізації його продуктивності чи роботи в іншому контексті. У разі, коли розробники хочуть перейти на нову мову програмування, транскompіляція може допомогти у перетворенні існуючого коду, щоб уникнути написання з нуля.

Великі проєкти можуть використовувати декілька мов програмування в різних частинах системи. Транскompіляція може полегшити спільну розробку і підтримку таких проєктів. Транскompіляція може використовуватися для внесення особливих функцій чи можливостей однієї мови в іншу.

Таким чином, обрана тема магістерської роботи є актуальною. У роботі розглянуто можливість дати спеціалістам такий інструмент, що дозволить працювати в комфортній для них середі-мові та використовувати код один одного.

Оскільки загальна система машинного перекладу досить складна, то в рамках дипломного проєкту приділено основну увагу розробці блоку кодогенерації, тобто системі автоматичної генерації коду на мові високого рівня.

Мета і завдання дослідження. Мета магістерської роботи - спрощення розробки застосунків, які мають працювати на багатьох платформах, за допомогою використання системи машинного перекладу з однієї мови високого рівня на іншу.

Для досягнення поставленої мети необхідно виконати такі основні завдання:

1. Провести аналіз підходів оптимізації мультиплатформеної розробки;
2. На базі проведеного аналізу визначити вимоги до розроблюваної системи автоматичної генерації коду;
3. Спроектувати систему автоматичної генерації коду;
4. Реалізувати систему автоматичної генерації коду;
3. Впровадити систему у компанію з розробки програмного забезпечення;
4. Провести аналіз отриманих результатів.

Об'єкт дослідження – синтаксис мов програмування високого рівня.

Предмет дослідження – система автоматичної генерації коду.

Методи дослідження базуються на синтаксисі мов програмування високого рівня. Мова Kotlin використовується для опису та розробки системи автоматичної генерації коду, система автоматизації збірки Gradle, XML формат для передачі проміжних результатів між різними етапами виконання генерації, середа розробки IntelliJ Idea, система керування версіями файлів Git, GitHub - репозиторій для вихідного коду.

Наукова новизна отриманих результатів:

розроблено та запропоновано архітектуру системи кодогенерації. Для подальшого масштабування вона має розділення на різні етапи обробки вхідних описів. Проміжні результати між

етапами зберігаються у вигляді дерев, таких як абстрактне синтаксичне дерево, дерево розбору, стилістичне дерево.

Практичне значення отриманих результатів:

реалізовано систему автоматичної генерації коду на мові високого рівня, яка з опису на метамові може генерувати структури та інтерфейси (примітиви) на мовах Kotlin, C/C++, Java, Swift з мінімальним втручанням людини. Кодогенератор викладено у вільний доступ для ознайомлення на платформі GitHub. Будь хто може використовувати розроблену систему у своїх проєктах по ліцензії GPL- 3.0. На даний час кодогенератор використовується для генерації коду у виробничому процесі для суміщення старої та нової частини комерційного продукту, написаного на різних мовах програмування, що допомагає спростити розробку та зекономити час розробників. Представлено акт про впровадження у виробничий процес ТОВ “ЛОГІТЕК ЮКРЕЙН” результатів науково-дослідної магістерської роботи.

Апробація результатів магістерської роботи. Основні положення магістерської роботи та результати досліджень подано до участі на конференціях:

- Шкрябець В.І. Транскомпілятор як засіб мультиплатформеної розробки / В.І. Шкрябець, Т.С. Дьячук // Матеріали XI міжнародної науково-практичної конференції «Сучасні проблеми і досягнення в галузі радіотехніки, телекомунікацій та інформаційних технологій», 12-14 грудня 2022 року, Запоріжжя. – 2022. – С. 119-121

- Шкрябець В.І. Система машинного перекладу з однієї мови високого рівня на іншу/ В.І. Шкрябець, Т.С. Дьячук // Тиждень науки-2023. Факультет комп'ютерних наук і технологій. Тези доповідей науково-технічної конференції, Запоріжжя, 24-28 квітня 2023 р. [Електронний ресурс] / Редкол. : Вадим ШАЛОМЄЄВ (відпов. ред.) Електрон. дані. – Запоріжжя : НУ «Запорізька політехніка», 2023. – С. 58-60

Структура та обсяг роботи. Магістерська робота складається зі вступу, чотирьох розділів, висновків, списку використаних джерел, додатку. Основна частина містить 71 сторінок, 19 рисунків і 1 таблиці, список використаних джерел зі 33 найменувань.

ОСНОВНИЙ ЗМІСТ РОБОТИ

У **першому розділі** проведено аналіз підходів для оптимізації мультиплатформеної розробки, таких як загальні мови програмування, скрипти, фреймворкі, методи кодогенерації та транскompіляції. У кожного з методів виявлені переваги, недоліки та області використання.

У результаті аналізу існуючих підходів, для подальшої розробки було сформовано вимоги до розроблюваної системи.

У **другому розділі** описано структуру, архітектуру та принцип роботи системи, що розробляється.

Оскільки різні мови програмування мають різну структуру проєктів та наповнення файлів з програмним кодом, то неможливо безпосередньо зробити переклад з однієї мови на іншу та отримати результуючий працюючий код. Цей процес має відбуватись поетапно та наведено на рисунку 1.

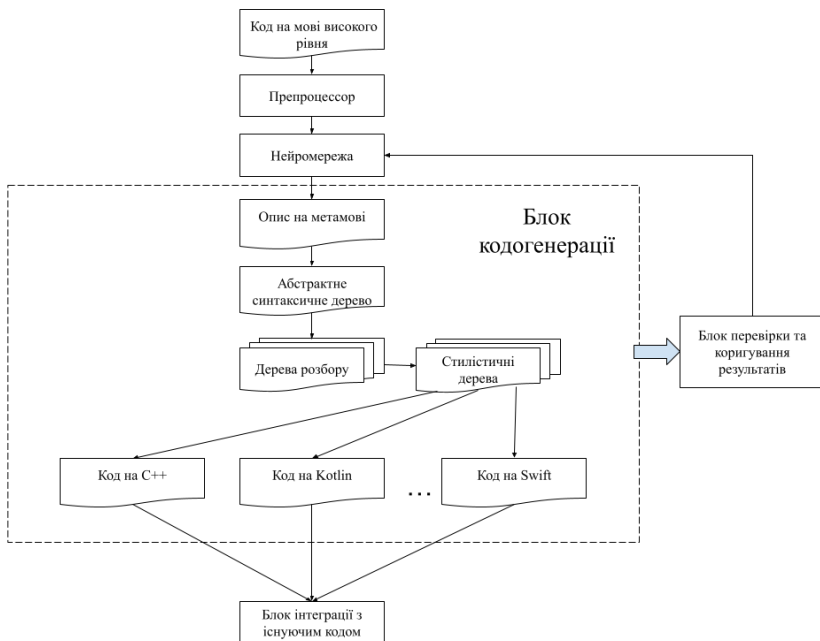


Рисунок 1 – Діаграма роботи системи машинного перекладу

Оскільки загальна система машинного перекладу досить складна, то в рамках роботи основна увага приділена блоку кодогенерації, тобто системі автоматичної генерації коду на мові високого рівня.

Блок кодогенерації починає роботу з обробки опису потрібного коду на метамові. Метамова описує загальну структуру програми та базується на Kotlin Script. Вона була розроблена в процесі проєктування системи та являється проміжним етапом для відладки.

Метамова містить засоби для декларативного опису структур, класів, інтерфейсів, типів даних, методів, тощо та дозволяє програмісту вивчати і змінювати опис структур в найбільш зручний для людини спосіб.

На основі опису на метамові система будує абстрактне синтаксичне дерево (Abstract Syntax Tree або AST). AST є структурою даних, яка представляє синтаксичну структуру програми, відображаючи її синтаксичний аналіз у вигляді дерева. AST є інтермедіативним представленням програмного коду, яке дозволяє відокремити синтаксичну структуру від конкретних деталей синтаксису. Кожен вузол в синтаксичному дереві представляє конструкцію мови програмування (таку як оператор, вираз, об'єкт чи інше) і має дочірні вузли, які представляють його складові частини або аргументи. Дерево може бути використано для виконання різних видів аналізу програми.

На наступному етапі система будує дерево розбору з урахуванням особливостей обраної мови, але без врахування стилістичних налаштувань. Дерево розбору - це структура даних, яка представляє перетворення абстрактного дерева в дерево, що враховує особливості та правила граматики вихідної мови. В цьому дереві містяться синтаксичні елементи тексту та їх взаємозв'язки згідно з правил граматики.

Дерево розбору не враховує можливі налаштування стандарту оформлення коду на мові високого рівня. Для цього потрібен окремий етап трансформації дерева розбору в стилістичне дерево, яке містить елементи оформлення коду, такі як табуляція, коми, переноси рядків, елементи коментування, інші спеціальні символи. Таким чином стилістичне дерево – це дерево розбору доповнено вузлами оформлення коду.

За стилістичним деревом система пише вихідні файли на обраній мові. В подальшому ці файли можна буде використовувати в проєктах користувача системи, тобто це і є фінальний результат системи кодогенерації.

У **третьому розділі** описано створення системи та основні проблеми, вирішення яких відбулося під час створення та реалізації системи.

Розроблено сценарії для побудови абстрактного синтаксичного дерева (рисунок 2), трансляції його в дерево розбору (рисунок 3), сценарії для додавання стилістичної інформації та запису в вихідні файли згенерованого коду. Для пояснення роботи системи наведені та розібрані приклади роботи трансляторів структур даних на мови C++ та Kotlin.



Рисунок 2 – Блок-схема сценарію побудови AST-дерева

Для початку роботи сценарію побудови абстрактного синтаксичного дерева потрібно завантажити налаштування проєкту (рисунок 2). Наступний крок - ітеративне проходження списку мов, які вказані в конфігурації проєкту в налаштуваннях.

Для кожної обраної мови буде виконано наступні кроки:

- завантажено та виконані скрипти на метамові;
- збережено результати виконання скриптів у вигляді XML дерева в зовнішні файли;
- мову для якої виконано всі кроки видаляємо з списку обробки.

Цей цикл повторюється для всіх вказаних в налаштуваннях мовах. Після чого алгоритм сценарію завершується. В результаті отримуємо XML - файл з абстрактним синтаксичним деревом.

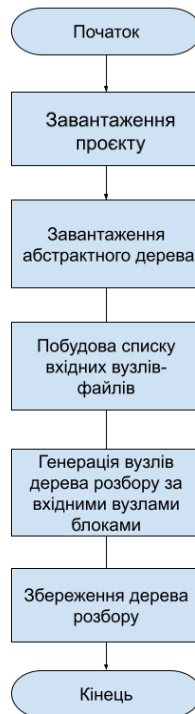


Рисунок 3 – Блок-схема сценарію побудови дерева розбору

Сценарій побудови дерева розбору для проєкту на вхід отримує шлях до абстрактного дерева, шлях до конфігурації проєкту та шлях куди зберігати дерево розбору (рисунок 3).

На початку роботи сценарію потрібно завантажити налаштування проєкту. Другий крок завантажити абстрактне дерево з вказаного файлу для вказаної в аргументах мови. На третьому кроці аналізуємо вхідне дерево, та будуємо список результуючих файлів за правилами прийнятими в обраній мові програмування. Четвертим кроком буде обхід абстрактного дерева, та трансляція ключових вузлів у відповідні структури обраної мови. Правила трансляції запрограмовані в коді для кожної мови та кожного виду блоків окремо. На останньому кроці дерево зберігається на диску у виді XML файлу.

На початку роботи сценарію побудови стилістичного дерева розбору потрібно завантажити налаштування проєкту. На другому кроці відбувається завантаження дерева розбору з вказаного файлу для вказаної в аргументах мови. На третьому кроці робимо обхід дерева розбору та створюємо копію дерева з додаванням стилістичних елементів. Єдиного алгоритму по додавання стилістичних елементів немає, тому система кодогенерації на цьому кроці приймає рішення в залежності від типу ключового вузла та обраної мови. На останньому кроці дерево зберігається на диску у вигляді XML файлу.

Останній сценарій системи кодогенерації - сценарій запису в файл. На початку роботи сценарію потрібно завантажити налаштування проєкту. На другому кроці відбувається завантаження стилістичного дерева з вказаного файлу для вказаної в аргументах мови. На останньому кроці робимо обхід стилістичного дерева та записуємо кожний вузол у відповідний вихідний файл. У вихідному файлу в результаті отримуємо згенерований програмний код.

У четвертому розділі розглянуто питання дослідження системи автоматичної генерації коду на мові високого рівня, а також практичне застосування такої системи.

Репозиторій з кодом системи кодогенерації викладено на GitHub за адресою <https://github.com/vshcryabets/codegen>. Система автоматичної генерації коду дозволяє генерувати загальні інтерфейси, блоки констант, перерахування та структури на мовах

Kotlin, C/C++ , Java, Swift. Планується додати підтримку інших мов високого рівня, зокрема Rust та Python. Ведеться робота з удосконалення трансляторів для генерації класів, методів та логіки. Також планується розробка методики взаємодії та поширення коду між командою розробників за допомогою інструментарію git. Система генерації коду під'єднується до системи збірки Gradle.

Галузь використання системи – компанії з розробки програмного забезпечення.

ВИСНОВКИ

Результати, отримані в магістерській роботі, є рішенням практичної задачі спрощення розробки застосунків, які мають працювати на багатьох платформах, за допомогою використання системи автоматичної генерації коду.

Отримано такі теоретичні та практичні результати:

Для досягнення поставленої мети необхідно виконати такі основні завдання:

1. Проведено аналіз підходів оптимізації мультиплатформеної розробки, на основі якого виявлено їх основні недоліки;

2. На базі проведеного аналізу визначено вимоги до розроблюваної системи автоматичної генерації коду;

3. Розроблена та запропонована архітектура системи кодогенерації. Для подальшого масштабування вона має розділення на різні етапи обробки вхідних описів;

4. Реалізовано систему автоматичної генерації коду. Система написана на мові Kotlin, та використовує XML формат для передачі проміжних результатів між різними етапами виконання генерації;

3. Систему викладено у вільний доступ для ознайомлення на платформі GitHub та впроваджено у виробничий процес ТОВ “ЛОГІТЕК ЮКРЕЙН”;

4. Проведено аналіз отриманих результатів. Кодогенератор з опису на метамові може генерувати структури та інтерфейси на мовах Kotlin, C/C++, Java, Swift з мінімальним втручанням людини. Робота над системою продовжується, репозиторій постійно оновлюється, виправляються помилки та система поліпшується.

Таким чином, всі поставлені в роботі задачі виконані й мета досягнута.