МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних
технологій

Кваліфікаційна робота бакалавра

на тему: Розробка програмного модуля перетворення програмного коду

Виконав студент групи К-19
спеціальності 122 Комп'ютерні науки
Довлетов Сердар

Керівник канд. техн. наук, доцент
Терещенко Т.М.

Рецензент док. техн. наук, проф.
Мещеряков В.І.

Одеса 2023

ЗМІСТ

ВСТУП

Конвертери програмного коду надають додаткові інструменти та бібліотеки, які полегшують розробникам інтеграцію своїх програм із існуючими системами або навіть створюють абсолютно нові з нуля. Будь-який розробник, експерт чи новачок може використовувати ці інструменти без поглиблених знань програмування завдяки великій кількості навчальних посібників, доступних онлайн для найпопулярніших наборів інструментів. Загалом це дає розробникам більше часу, щоб зосередитися на написанні кращих програм, а не витрачати час на спроби зрозуміти незнайомий синтаксис або структури даних.

Конвертери програмного коду – це універсальне програмне забезпечення, яке дозволяє розробникам легко переміщувати свої програми між платформами без необхідності починати з нуля. Ці інструменти стали популярними як серед програмістів-професіоналів, так і серед програмістів-аматорів завдяки низькій вартості, широкому набору функцій і величезній спільноті підтримки.

Конвертери залучили більше користувачів, оскільки вони дозволяють розробникам отримувати доступ, змінювати та розширювати свій код. Це дозволяє їм швидко та легко адаптуватися до нових технологій, а також створювати індивідуальні рішення. Також конвертори з відкритим програмним кодом забезпечують широке охоплення, ніж програмне забезпечення із закритим кодом, дозволяючи розробникам ділитися своїм кодом зі світом. Це полегшує поширення розробки коду, що може призвести до нових ідей та інновацій. Розробники можуть додавати нові функції та можливості, які інакше були б недоступні в закритому програмному забезпеченні. Це дозволяє швидше створювати та розгортати більш складні програми.

Метою роботи є розробка програмного модуля перетворення програмного коду.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Визначити напрямки використання конверторів програмного коду в процесах розробки програмного забезпечення для різноманітних задач.

2. Проаналізувати існуючі програмні інструменти перетворення коду, особливу увагу приділити функціональним особливостям та готовим конверторам програмного коду.

3. Розробити алгоритм перетворення програмного коду з урахуванням обмежень, які накладаються за рахунок відмінностей синтаксису мов програмування.

4. Провести огляд обмежень розробленого алгоритму та визначити шляхи подолання цих обмежень за рахунок використання додаткових інструментів або програмних рішень.

5. Розробити програмні інструменти для перетворення програмного коду.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Основні функції конверторів коду

Конвертери програмного коду – це особливий тип програмного забезпечення, яке перекладає одну мову програмування на іншу. Його можна використовувати в багатьох різних контекстах, таких як розробка ігор, веб-дизайн і розробка операційної системи. Конвертери програмного коду дозволяють розробникам легко переносити свої програми з одного середовища в інше, завдяки чому створювати ту саму програму на кількох платформах набагато швидше та легше.

Більшість конвертерів написані на мові програмування C завдяки її гнучкості та ефективному часу обробки. Це робить його ідеальним для перекладу між двома мовами, які можуть мати дуже різний синтаксис або типи даних. Багато з цих інструментів підтримують перетворення як компільованих мов (таких як C++ або Java), так і інтерпретованих мов (наприклад, Python або JavaScript). Інші функції включають автоматично створену документацію та параметри форматування для забезпечення точності під час рефакторингу коду з однієї мови на іншу [1].

Сучасний ринок програмного забезпечення пропоную як комерційні продукти, так і інструменти з відкритим програмним кодом. Основні переваги використання інструменту з відкритим вихідним кодом включають економію коштів завдяки уникненню плати за комерційні ліцензії, а також доступ до величезної спільноти, яка займається швидкою та ефективною розробкою передових технологій без додаткових витрат. Крім того, в Інтернеті зазвичай доступний широкий спектр безкоштовних ресурсів щодо прикладів використання, порад і підказок, посібників з усунення помилок, які допомагають будь-кому швидко розпочати роботу з кодування, незалежно від рівня досвіду чи технологічних навичок будь-якого програміста.

Основні функції, які реалізовані в будь-якому конверторі програмного коду (рис. 1):

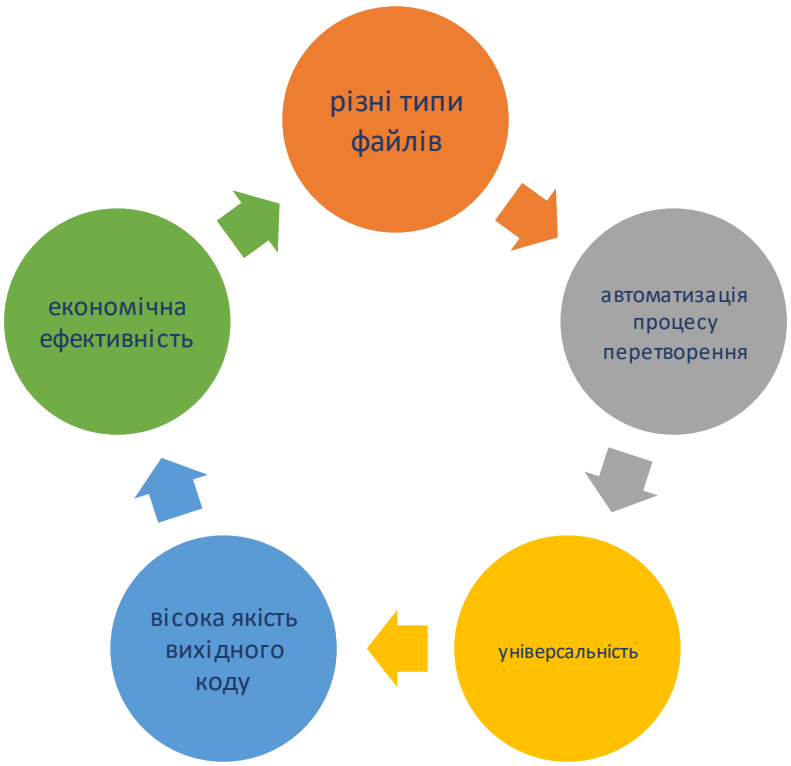

Рисунок 1 – Основні функції конверторів програмного коду

Розглянемо кожну зазначену функцію більш докладно:

1. Перетворення різних типів файлів: конвертери надають можливість конвертувати різні типи файлів, наприклад C, C#, C++, HTML, XML і Java, у різні формати. Це гарантує, що користувачі можуть створити надійний набір вихідних файлів швидше, ніж за допомогою ручних процесів.

2. Можливість автоматизувати процеси перетворення коду: конвертери пропонують функції, які дозволяють користувачам автоматизувати процес перетворення коду, тому їм не потрібно вручну конвертувати кожен тип файлу окремо. Це допомагає заощадити час і енергію, одночасно забезпечуючи точність вихідних файлів.

3. Універсальність: завдяки своїй універсальності конвертери дають користувачам можливість адаптувати їх для власних проектів або

організаційних стандартів. Вони також здатні ефективно обробляти масштабні проекти з мінімальними зусиллями користувачів завдяки своїй масштабованості та модульності.

4. Вихідні дані високої якості: конвертери зазвичай створюють високоякісні вихідні формати, які відповідають галузевим стандартам або специфікаціям організації. Завдяки цьому компанії можуть зменшити ризик потенційних помилок або помилок, які проникнуть у кінцевий продукт перед його випуском у виробництво.

5. Економічна ефективність: існують безкоштовні конвертери, що робить їх дуже бюджетними для компаній з обмеженим IT-бюджетом, яким може знадобитися доступ до якісних результатів без шкоди стандартам якості та результатам.

1.2. Напрямки використання конверторів коду

Різноманіття програмних продуктів такого типу обумовлено широким колом завдань та користувачів, які ці завдання виконують. Слід зазначити, що існує безліч подібних програмних інструментів, наприклад (рис. 2):

1. Перетворювачі тексту в HTML: вони перетворюють звичайний текст у код HTML, створюючи готовий макет веб-сторінки.

2. Конвертери Excel у XML/JSON: вони дозволяють конвертувати необроблені дані, що зберігаються в електронних таблицях Microsoft Excel, в інші формати, наприклад XML або JSON.

3. Транскодери аудіофайлів: вони використовуються для перетворення форматів аудіофайлів в інші аудіофайли, включаючи методи стиснення без втрат і з втратами.

4. Конвертори форматів зображень: ці інструменти дозволяють користувачам перетворювати файли зображень з одного типу в інший, наприклад із TIFF у JPEG або PNG.

5. З'єднувачі баз даних: з'єднувачі баз даних доповнюють розрив між різними системами керування базами даних (СУБД), дозволяючи передавати дані з однієї системи в іншу.

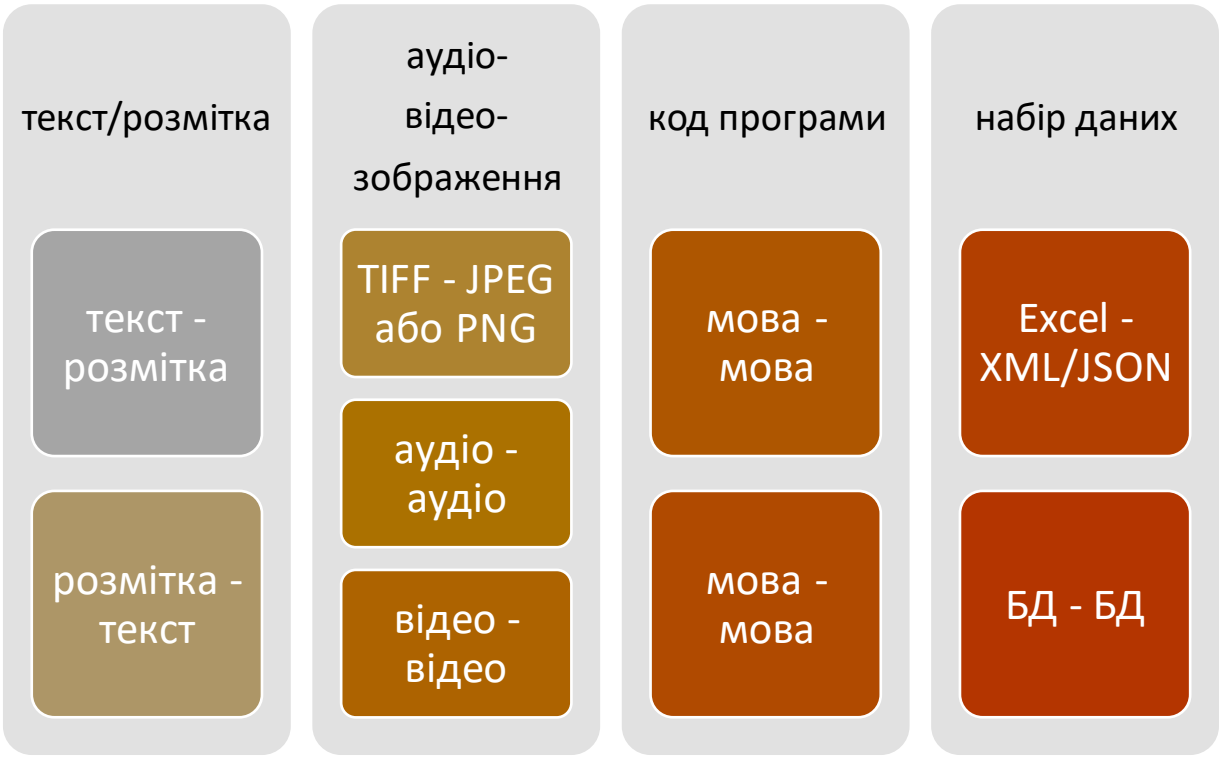

Рисунок 2 – Типи конверторів програмного коду

6. Відеокодеки: дозволяють конвертувати відеофайли різних форматів (MP4, AVI, MOV тощо) для відтворення на різних програвачах і пристроях.

7. Парсери мови розмітки: вони дозволяють читати, обробляти та перетворювати необроблений код розмітки (XML, HTML тощо) у потрібний формат.

8. Генератори компіляторів: Генератори компіляторів беруть вихідний код, написаний однією мовою програмування, і перетворюють його на іншу мову, яка потім може бути інтерпретована машиною.

Розгляне перш за все переваги, які надають конвертери програмного коду з відкритою ліцензією, а саме:

1. Підвищена доступність: конвертери надають розробникам легший і швидший доступ до найновіших стандартів мов програмування, гарантуючи, що їхні проекти залишаються сучасними та актуальними.

2. Економія: використовуючи конвертери відкритого коду замість дорогих інструментів розробки, команди можуть заощадити багато грошей на дорогих ліцензіях і зборах.

3. Покращена якість: особливо конвертери з відкритим вихідним кодом ретельно перевіряються спільнотою, що призводить до зменшення кількості помилок, зберігаючи кращу якість порівняно з традиційними процесами розробки.

4. Просте обслуговування: за допомогою конвертера набагато простіше оновлювати або змінювати код, коли це необхідно. Це означає, що витрати на підтримку та технічне обслуговування значно зменшуються порівняно з альтернативами із закритим кодом.

5. Гнучкі параметри використання: відкриті коди пропонують розробникам більше гнучкості у використанні програмного забезпечення, дозволяючи їм налаштовувати або розширювати функціональність залежно від своїх потреб.

6. Переваги безпеки: використання перетворення коду з відкритим вихідним кодом також допомагає захистити розробників від проблем із безпекою, оскільки всі оновлення контролюються спільнотою, тому потенційним загрозам важче пройти непоміченими.

7. Підтримка найновіших стандартів: відкриті коди дозволяють розробникам швидко приймати нові мовні стандарти, не витрачаючи надто багато часу чи зусиль на вивчення абсолютно нової мови з нуля.

Визначимо типи користувачів, які використовують конвертери відкритого коду (рис. 3).

Бізнес-користувачі: користувачі, які прагнуть підвищити потужність бізнесу чи іншого комерційного підприємства, можуть використовувати для своїх операцій конвертери з відкритим кодом. Вони можуть використовувати

рішення з відкритим кодом для веб-розробки, проектування програмного забезпечення або системної інженерії.

Професіонали: інженери-програмісти, дизайнери, інформатики та інші професійні спеціалісти з IT можуть скористатися конвертерами відкритого коду, використовуючи їх для розробки складних рішень, які відповідають їхнім конкретним вимогам і потребам.

Рисунок 3 – Типи користувачів конверторів програмного коду

Студенти: Студенти, які вивчають інформатику, розробку програмного забезпечення та суміжні теми, можуть використовувати конвертери відкритого коду як доступний спосіб досліджувати різні типи систем кодування та технологій без необхідності інвестувати кошти у дорогі засоби розробки.

Викладачі: Викладачі, зокрема вчителі та професори, часто використовують конвертери відкритого коду як навчальні посібники, щоб

ефективніше продемонструвати різні концепції мов кодування та принципів програмування.

Користувачі типу DIYers: Непрофесійні користувачі технологій, відомі як аматори або ентузіасти Do It Yourself (DIY), також складають значну частину тих, хто використовує конвертери відкритого коду для створення проектів зі ступенем складності, якого вони не зможуть досягти. інакше.

Дослідники. З наукового боку дослідники в таких галузях, як штучний інтелект (ШІ), машинне навчання (МН) і робототехніка, також використовують рішення з відкритим кодом, щоб економічно ефективно досягати своїх дослідницьких цілей.

Некомерційні організації: такі організації часто використовують конвертери з відкритим вихідним кодом для підтримки своєї роботи, особливо у випадках, коли організації потрібно швидко розробити рішення без великих витрат на розробку.

Підприємці: підприємці-початківці та власники бізнесу, зокрема, отримують вигоду від конвертерів відкритого вихідного коду як способу створення власних продуктів і послуг більш економічно.

Урядові установи: Уряди також часто звертаються до рішень з відкритим кодом для проектів розробки програмного забезпечення, щоб зменшити витрати та отримати доступ до новітніх технологій без сплати ліцензійних зборів.

Конвертери з відкритим вихідним кодом зазвичай безкоштовні. Це пояснюється тим, що конвертери з відкритим вихідним кодом розробляються програмістами-добровольцями, які зосереджуються на створенні програмного забезпечення та інструментів, які допомагають розробникам створювати програми та веб-сайти швидше й ефективніше. Крім того, конвертери з відкритим вихідним кодом часто поширюються з ліцензією, яка дозволяє користувачам використовувати програмне забезпечення будь-яким способом, у тому числі змінювати або розповсюджувати оригінальну роботу на інших умовах. Крім того, через процес розробки конвертери з відкритим кодом часто

оновлюються виправленнями помилок і оновленнями безпеки, що підвищує їх цінність як надійного інструменту для розробників. Таким чином, конвертери з відкритим вихідним кодом надають неоціненну послугу розробникам, не коштуючи їм взагалі нічого.

Основна перевага використання саме конверторів відкритого коду полягає у зменшенні витрат на розробку та супровід програмного забезпечення. Як показав аналіз типів користувачів, такі програмні інструменти мають широке коле користувачів, як фізичних, так і юридичних. Але не слід забувати, що не тільки знижені матеріальні витрати є превагою для таких конверторів, великим «плюсом» використання є наявність широкої професійної спільноти.

Наступне важливе питання – це типи програмного забезпечення, які інтегруються конвертери програмного коду як комерційні, так і некомерційні. Конвертери можуть інтегруватися з різними типами програмного забезпечення, включаючи операційні системи, засоби розробки та браузери. Такі операційні системи, як Windows і Linux, частіше забезпечують підтримку програм перетворення коду з відкритим вихідним кодом, що дозволяє використовувати їх для програм системного рівня. Інструменти розробки, такі як IDE (інтегроване середовище розробки), як правило, мають вбудовану підтримку утиліт перетворення коду обох типів ліцензії, що робить процес впровадження перетвореного коду безперебійним. Нарешті, багато сучасних веб-браузерів постачаються зі складними двигунами JavaScript, які можуть інтерпретувати та запускати відкритий вихідний код безпосередньо з самого браузера, забезпечуючи простий спосіб виконання сценаріїв, написаних різними мовами.

2. ПРОГРАМНІ ІНСТРУМЕНТИ ПЕРЕТВОРЕННЯ КОДУ

2.1. Функціональні особливості конверторів мов програмування

Перетворення коду з однієї мови програмування на іншу полягає не тільки у перетворенні синтаксису між мовами програмування. Це процес перетворення даних зі збереженням структури, наскільки це можливо. Нещодавні дослідження показують, що міграція існуючої кодової бази на більш сучасну чи ефективнішу мову, таку як Java або C++, потребує навичок як вихідної, так і кінцевої мов і може бути дорогим. Наприклад, Австралійський банк Співдружності витратив приблизно 750 мільйонів доларів протягом п’яти років, щоб перевести свою платформу з COBOL на Java. Слід зазначити, що транскомпілятор може заощадити час шляхом усунення необхідності переписувати код з нуля. Проте його складно реалізувати на практиці, оскільки різні мови мають різний синтаксис і покладаються на різні API платформи , стандартні бібліотечні функції та типи змінних [2].

Обмеження перенесення застарілого коду на іншу мову програмування зустрічаються на різних рівнях, кожен з яких є більш амбітним і важчим для реалізації. Як результат, у міру просування по рівнях потреба в ручному перетворенні зростає. На найпростішому рівні міграція стоїть процес перетворення коду з однієї мови на іншу. На більш високому рівні – структура системи може бути змінена, наприклад, шляхом перетворення коду, написаного на суто процедурній мові, у код, написаний на об'єктно-орієнтованій мові. На більш високих рівнях може знадобитися змінити глобальну архітектуру. Підтримувати узгодженість між різними мовами стає все важче, оскільки щодня з’являються нові технології. З розвитком мов програмування також з’являються нові функції. У результаті будь-які компілятори перетворення, пов’язані з цією мовою, повинні бути оновлені.

Наприклад, George зазначив, що кожні 3-5 років ANSI C++ оновлюється. До зміненого формулювання можуть бути включені нові ключові слова. Такі як constexpr, яке є новим ключовим словом у ANSI C++0x . У результаті будь-які ідентифікатори з однаковими іменами в ранніх програмах потрібно оновити, інакше вони будуть неправильно розтлумачені.

Розглянемо деякі програмні інструменти, які доступні для перекладу коду між мовами програмування. Їх досить багато, але частіше згадуються наступні:

JLCA – Java Language Conversion Assistant – це інструмент, який автоматично перекладає поточний код Java у код Visual C# .

BCX – це невеликий інструмент командного рядка, який бере файл вихідного коду BCX BASIC і перетворює його на файл вихідного коду C, який можна скомпілювати будь-яким компілятором C або C++.

PERTHON – перекладає вихідний код Python у вихідний код Perl 5.x , який розробники можуть читати. Він аналізує за допомогою Parse::Rec Descent Деміана Конвея та намагається повторно реалізувати мову Python відповідно до Довідкового посібника Python і граматики BNF.

GWT від Google – за допомогою GWT можливо створювати та налагоджувати програми AJAX на Java за допомогою бажаних інструментів розробки Java. Компілятор GWT перетворює готову програму Java на сумісні з браузером JavaScript і HTML.

Hiphop від Facebook – HipHop компілює вихідний код PHP за допомогою g++ після програмного перетворення його на високоефективний C++.

TransCoder AI від Facebook – це система Facebook, яка може перекладати між C++, Java і Python, використовує підхід до задачі без нагляду. TransCoder починається з міжмовної моделі попереднього навчання, яка перетворює фрагменти коду, що виражають однакові інструкції, в еквівалентні представлення незалежно від мови програмування.

Метою програмістів є досягнення максимальної ефективності перетворення без шкоди для якості перетвореної системи. Незважаючи на те, що перетворення мови здається простим, це важке завдання з численними складнощами. Його належним чином вважають однією з десяти найбільших труднощів програмування, з якими стикаються розробники всього світу. Ще до того, як стане доступним надійний напівавтоматичний перетворювач, необхідно досягти значного прогресу. Можливо, у майбутньому стане можливим перекладати код між двома абсолютно різними платформами одним клацанням миші. Оскільки бібліотека сама по собі не може бездоганно виконати таке перетворення та виявити шаблон заміни в мові, це може залежати від прогресу в області штучного інтелекту. Хорошим прикладом є перетворення консольної програми у веб-додаток і навпаки.

2.2. Обмеження перетворення коду мейнфрейму

Задання перетворення коду мейнфрейму на іншу мову програмування є доволі актуальним. Це пов’язано не тільки із розвитком індустрій програмування, а й зі скороченням фахівців здатних якісно підтримувати такі системи.

Існують деякі важливі застереження, пов’язані з використанням автоматизованих інструментів для перетворення коду, написаного такими мовами, як Fortran і COBOL, у сучасні. Процес перетворення майже завжди вимагає більше роботи, ніж просто натискання кнопки, і є питання безпеки та керування кодом, про які також варто подумати. Стаття [3] розкриває ці виклики, щоб пояснити, які інструменти перетворення коду мейнфрейму можуть, а які ні, і коли має сенс, а коли їх використовувати. Розглянемо основні аспекти цього процесу, які були досліджені автором.

Перш за все треба визначити основне поняття цього процесу. Автоматизоване перетворення коду мейнфрейму – це процес автоматичного перетворення коду, написаного на «застарілій» мові, такій як Fortran або

COBOL, на «сучасну», таку як Java або C. Визначення «застаріла» та «сучасна» доволі умовні, оскільки не кожен мейнфрейм-додаток потрібно переписувати новою, сучаснішою мовою. Але іноді виникають завдання конвертувати старішу мейнфрейм-програму на більш сучасну мову. Наприклад, компанія може більше не мати в штаті програмістів, які знають, як підтримувати застарілий код, або вона має бажання використовувати інструменти розробки чи спостереження, які не підтримують старіші мови. Переформатування або повторне розміщення додатків мейнфрейму також може вимагати перетворення коду на нову мову [3].

У подібних ситуаціях автоматизовані засоби перетворення коду мейнфрейму можуть заощадити розробникам багато часу та людського ресурсу. Завдяки автоматичному перетворенню коду вони усувають необхідність переписувати програми з нуля, рядок за рядком, новою мовою. Адже цей процес може зайняти роки, якщо він передбачає велику кодову базу. Крім того, інструменти автоматичного перетворення можуть знизити рівень досвіду, необхідного для виконання перетворення коду, оскільки розробникам зазвичай не потрібно володіти як вихідною, так і цільовою мовами для використання інструментів.

Ще одна перевага автоматичного перетворення коду мейнфрейму полягає в тому, що воно, як правило, створює більш узгоджений код, ніж ручний процес перетворення. Якщо попросити кількох розробників конвертувати код вручну, швидше за все, на виході отримаємо різні стилі коду в переписаній програмі, оскільки кожен програміст працюватиме по-різному. Але за допомогою автоматичного інструменту перетворення весь код перетворюється однаково.

Сьогодні доступні різноманітні інструменти для перетворення коду, орієнтовані на програмне забезпечення мейнфреймів. Деякі з них безкоштовні та з відкритим кодом. Наприклад, конвертер Fortran-C++ і Fortran-to-Julia є на GitHub. Ці інструменти легкодоступні, і вони не коштують нічого, але вони можуть мати ряд недоліків та помилок. Такі компанії, як Modern

Systems і Asysco, пропонують комерційні інструменти перетворення, які, як правило, є трохи зручнішими для користувача. Деякі з них також підтримують кілька вихідних і цільових мов, що зазвичай не стосується інструментів конвертації з відкритим кодом.

Завдання визначення на першому етапі якості майбутнього перетворення програмного коду того чи іншого інструменту є актуальною для будь-якої компанії чи окремого користувача. У деяких випадках самі розробники інструменту попереджають, що код, створений за допомогою автоматичного перетворення, «може компілюватися або не компілюватися», як зазначають автори конвертера Fortran-to-C++ . І навіть якщо код усуває мінімальну перешкоду для компіляції, він може містити проблеми з безпекою та, ймовірно, не оптимізований для досягнення найкращого можливого перетворення.

Комерційні рішення для перетворення, як правило, пропонують більш обнадійливі обіцянки щодо якості коду, який вони генерують. Але частково це тому, що постачальники, які стоять за цими інструментами, продають не лише самі інструменти, але й ручний процес, за допомогою якого вони допомагають клієнтам спланувати та налаштувати процес перетворення перед фактичним його виконанням. Завчасне планування та пристосування процесу перетворення до конкретних потреб відповідного проекту допомагає отримати кращий результат. Але це також збільшує час, необхідний для завершення процесу перетворення, і зусилля, які розробники мають докласти до цього.

Визначимо умови, за яких слід використовувати програмні інструменти автоматичного перетворення коду майнфреймів. До основних слід віднести наступні:

1. Потреба конвертувати багато коду, принаймні десять тисяч рядків або сотні тисяч, конвертувати такий код вручну просто не практично.

2. Якість написання вхідного коду дуже висока. Чим краща якість існуючого коду, тим краще інструменти перетворення переведуть його на іншу мову програмування.

3. Недостатній досвід програмування. Це стосується і мов мейнфреймів, і сучасних мов.

4. Вагома причина перетворення існуючого коду. Продовжувати підтримувати його на «застарілій» мові просто не має сенсу.

З іншого боку, автоматичне перетворення коду слід уникати у наступних умовах:

1. Програма проста або маленька, її легко переписати вручну.

2. Програма має відповідати особливо суворим вимогам до продуктивності та безпеки.

Слід зазначити ще одне, автоматичне конвертування коду не виключає перевірки та налаштування його згодом, іноді суттєво, щоб переконатися, що він ефективний і безпечний. Перетворення мейнфрейм-додатків не є простою та одноразовою дією. Автоматизовані інструменти можуть заощадити багато часу та зусиль, але це зовсім не означає, що автоматичне перетворення є процесом, який не потребує зусиль.

2.3. Аналіз існуючих трансляторів програмного коду

Транслятор мови кодування – це інструмент, який перекладає код програми з вихідного коду в код призначення. Переклад мов кодування схожий на той самий процес перекладу мов з однієї на іншу, але є більш складним. У процесі роботи змінюються дані, але виникає необхідність зберегти структуру даних, наскільки це можливо. Теоретично фрагмент коду можна перекласти на будь-яку іншу мову, якщо мови програмування мають однаковий параметр «повнота за Тьюрингом». Це властивість у теорії обчислюваності, за якої можливе виконання будь-яких обчислень будь-яким іншим методом обчислення. Компілятори використовуються для виконання перекладів мов програмування, але компілятори часто використовуються для зміни коду, щоб машина могла його інтерпретувати. Він вимагає підмножини

компіляторів, які називаються транспілерами, щоб код був зрозумілим для людини. Транспілер (компілятор від джерела до вихідного коду) – це транслятор, здатний здійснювати переклад між мовами програмування, які працюють на однаковому рівні абстракції. На жаль, створити транспілер на практиці складно, оскільки різні мови мають різний синтаксис і покладаються на різні API платформи та стандартні бібліотечні функції.

Переклад мов програмування є складним завданням, яке включає збереження точної семантики виконання та загальну подібність коду одночасно. Процес ускладнюється функціональними можливостями програмного забезпечення або їхньою відсутністю, і особливостями мов програмування. Наприклад, програма має багато зовнішніх залежностей, які важко замінити в цільовому технологічному стеку. Процес перетворення такого коду буде значно відрізнятися від перекладу програми, яка реалізує все всередині. Ще приклад, коли у вихідному чи цільовому коді може бути кілька субтитрів, які потрібно враховувати або вони є такими, які не мають значення для перекладу. Зрозумілим прикладом є операція «+» у Python, C та C#. Операнд є тією самою операцією, але виконується по-різному. Тобто, математично додавання є правильним у Python, але натомість може розширюватися більше ніж на 32 біти, воно залишається до 32 бітів лише в C, а в C#, це може викликати виняток залежно від режиму. Наступна складність – виклик бібліотечних функцій. Коли перекласти код між мовами програмування з різною семантикою. Наприклад, з Python на C#, функції Python у коді не будуть працювати у C#, оскільки вони в ньому недоступні. З часом багато дослідників прийшли до ідеї перекладу мов кодування. Розглянемо де-які з інструментів для перекладу мов кодування [4].

1. TransCoder AI від Facebook. TransCoder AI – це система перекладу мов програмування, розроблена дослідниками Facebook, яка може перекладати код між мовами C++, Java та Python. Він заснований на неконтрольованому алгоритмі машинного навчання для виконання перекладу та є одним із найкращих транспіляторів Python. Він пройшов навчання на

більш ніж 2,8 мільйонах проектів з відкритим кодом і перевершив існуючі системи перекладу коду за допомогою методів перекладу на основі правил. Алгоритм, який використовується в TransCoder, створено під впливом системи нейронного машинного перекладу (NMT) на мови програмування. Алгоритм TransCoder визначає загальні елементи між мовами введення та виведення, які називаються токенами. Токени можна визначити як ключові слова в мовах програмування, як-от «for», «if», «else», «try» тощо, а також математичні цифри та оператори. Деякі маркери є звичайними рядками, які присутні в коді. Потім якість перекладу покращується за допомогою алгоритму з використанням методу зворотного перекладу. Метод зворотного перекладу спонукає до створення вихідного коду до кінцевого коду та цільового до вихідного коду одночасно та з'єднується разом у кінці, щоб отримати остаточний результат.

Тестування на точність проводилося за допомогою 852 паралельних функцій на C++, Java та Python від GeeksforGeeks. Точність обчислення було розраховано під час перекладу між C++, Java та Python на основі пошуку BEAM із N=25. У результаті було отримана наступна обчислювальна точність перекладу:

- C++ в Java – 74,8%
- C++ до Python – 67,2%
- Java до C++ – 91,6%
- Java до Python – 68,7%
- Python до Java – 56,1%
- Python до C++ – 57,8%.

TransCoder працює на базі трансформатора архітектури від послідовності до послідовності, що складається з кодера та декодера. Він дотримується трьох принципів: одна міжмовна модель маскованої мови,

попереднє навчання, автоматичне кодування з усуненням шумів і зворотний переклад.

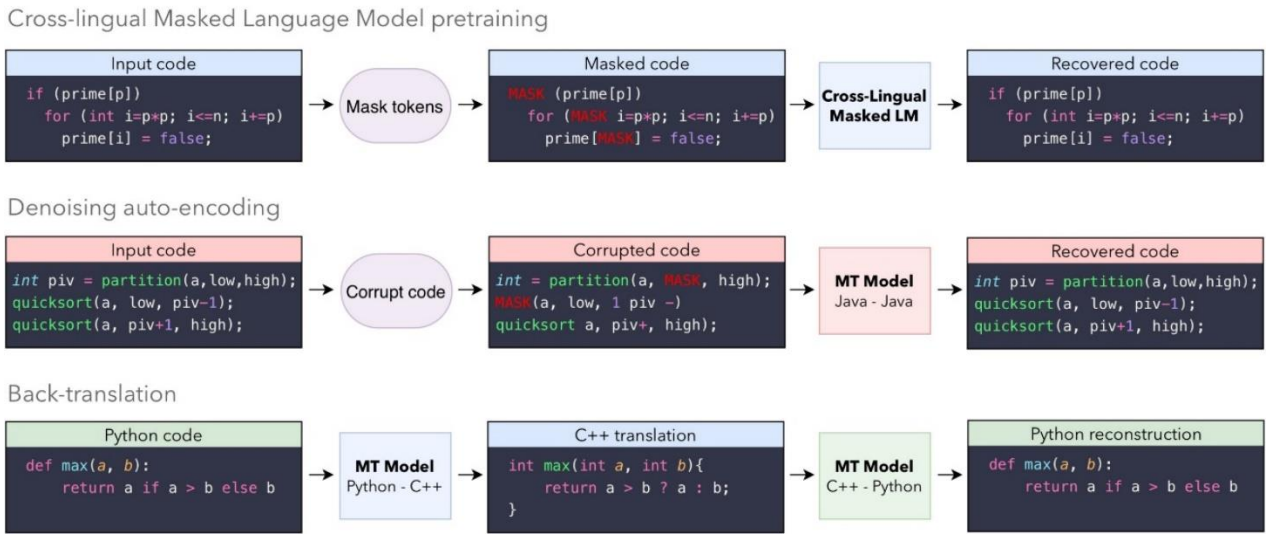

Рисунок 4 – Модель перетворення коду (джерело: arxiv.org)

2. CodeNet від IBM. CodeNet – це незавершений проект IBM, який має на меті навчити ШІ кодувати. На конференції IBM Think 2021 Арвінд Крішна, генеральний директор IBM, представив проект CodeNet, над яким д-р Ручир Пурі та дослідницька група IBM розробляли два роки. Проєкт CodeNet – це масштабний набір даних із приблизно 14 мільйонами зразків коду, написаних понад 50 мовами програмування. Широкий вибір мов і проблем кодування, які пропонує CodeNet, містить понад сотню рішень для кожного з 80% проблем. Ідея проєкту CodeNet полягає в тому, щоб дозволити дослідникам і розробникам допомагати в пошуку коду, доповненні коду, перекладі коду в код і комбінації інших випадків використання. CodeNet схожий на ImageNet, містить набір даних комп'ютерного зору, що містить 14 мільйонів зображень, розкиданих по 20 000 категоріях. Хоча робота CodeNet для перекладу коду в код ще не готова, це багатообіцяючий проєкт, який, як очікується, буде добре працювати.

3. GWT Google. GWT, набір веб-інструментів Google, – це набір інструментів розробки для створення та оптимізації складних програм на основі браузера. Він розповсюджується з відкритим вихідним кодом, який дозволяє розробникам створювати та підтримувати інтерфейсні програми JavaScrip. Мета GWT полягає в тому, щоб зробити можливим продуктивне створення високопродуктивних веб-додатків без необхідності розробникам бути експертом у JavaScript, XMLHttpRequest або особливостях браузера. GWT – це популярний перекладач мов кодування, який працює для створення та налагодження додатків AJAX у Java, а коли його використовують у процесі виробництва програмного продукту, компілятор перетворює програму Java у зручні для браузера JavaScript і HTML.

GWT складається з двох основних компонентів:

1) GWT SDK, який містить бібліотеки Java API, компілятор і сервер розробки. Це дозволяє користувачу писати клієнтські програми на Java та розгортати їх як JavaScript.

2) Плагін для Eclipse, який забезпечує підтримку IDE для веб-проектів GWT і механізму додатків.

4. GitHub Copilot. Це парний програматор штучного інтелекту на основі OpenAI Codex, нової системи штучного інтелекту, створеної OpenAI. Він допомагає розробникам писати код швидше з меншими зусиллями, пропонуючи пропозиції у стилі автозавершення під час кодування. Пропозиції надаються або безпосередньо під час написання коду, або шляхом написання коментаря природною мовою про те, що розробник хочете, щоб код робив.

Це схоже на надання команд для написання коду. GitHub Copilot надає різні варіанти використання кодування, зокрема написання коду за допомогою простої команди кількома мовами, створення словників пошукових даних, навігацію новою кодовою базою за допомогою GitHub Copilot Labs тощо.

Рисунок 5 – Процес перетворення коду (джерело:GitHub)

4a. GitHub Copilot Labs. Це супутнє розширення GitHub Copilot, яке можна встановити з Visual Studio. Це розширення коду візуальної студії, яке містить експериментальні програми Copilot, які є функціями редактора на основі ML для розробників. Однією з особливостей Copilot Labs є переклад мов кодування з однієї мови на іншу, яка включає близько 60 мов кодування на вибір.

Завдання перекладу мови коду є складним, і багатьом буде цікаво, коли воно стане реальністю. TransCoder від Facebook показує чудові результати, успішно розуміє синтаксис, специфічний для кожної мови, і вивчає структури даних та їх методи. На вищих рівнях підтримувати транспілер складніше, ніж розробляти його, оскільки з'являються нові технології та вводяться нові функції з розвитком мов програмування. Як наслідок, робота над транспілерами є послідовним процесом і потребує великих людських ресурсів. Саме це і є основними обмеженнями створення такого програмного продукту. Можливо подальший розвиток ШІ підштовхне вирішення цього питання у правильному напрямку.

Початок роботи з конвертерами відносно простий, однак перед початком роботи потрібно виконати кілька кроків. По-перше, користувачі повинні

визначити, у який тип кодування вони хочуть перетворити свій код. Конвертери зазвичай дозволяють конвертувати між кількома мовами програмування та мовами розмітки, тому найкраще вирішити, якій мові надає перевагу користувач і яка мова найкраще для проекту.

Після того, як користувач вибрав мову, яку він хоче використовувати, він повинен шукати онлайн-конвертер або програму, спеціально розроблену для цього завдання. Багато веб-сайтів мають безкоштовні інструменти для конвертації коду, і їх, як правило, легко знайти, виконавши простий пошук в Інтернеті. Деякі з цих інструментів перетворення можуть вимагати певних бібліотек або плагінів для належної роботи, тому важливо переконатися, що всі необхідні компоненти встановлено, перш ніж продовжувати.

Після встановлення всіх компонентів користувачі можуть почати конвертувати свій код з однієї мови на іншу. Залежно від того, наскільки великий вихідний документ і наскільки складний його синтаксис, цей процес може тривати від кількох секунд до кількох хвилин залежно від розміру файлу, який перетворюється. Після успішного перетворення коду слід протестувати кілька невеликих фрагментів щойно перетвореного сценарію як додаткову міру обережності перед впровадженням його у кінцевий продукт. Іноді під час перетворення можуть траплятися друкарські чи інші помилки.

3. АЛГОРИТМ ПЕРЕТВОРЕННЯ КОДУ

В роботі [5] запропоновано підхід, який використовується для вирішення проблеми перетворення однієї мови кодування в іншу. Цей методом є подібним до методу перерахування [6]. Це означає використання операторів if для виведення списку та виявлення всіх можливих структур, які існують у коді Python, і перетворення кожної частини коду у її версію Java. Після ведення файлу Python конвертер розбиває його рядок за рядком, для кожного рядка він швидко перетворює прості частини пробілів для відступів і коментарів, тому в перетворення переноситься лише значущий код. У певному порядку алгоритм перевіряє унікальні слова, які представляють список часто уживаних структур коду Python. Прикладом є використання знаку «=», який містить будь-який рядок коду. Це означає, що рядок містить змінну, тобто значення правого боку має бути збережено в лівому бік.

Для створення ефективного та раціонального алгоритму перетворення необхідно визначити порядок перетворення окремих елементів коду (рис. 6).

Це необхідно для того, щоб визначити структури, оскільки в одному рядку може бути кілька структур, наприклад оператор if, який перевіряє певне значення в списку за допомогою методу. Для структур, які не включені в цей алгоритм, наприклад, відкриття та читання файлів, інші цикли, інші функції, автори пропонують вилучити та конвертувати в коди Java, а інші частини рядків Python залишаються незмінними. Після того, як рядок Python вийде з усіх цих випадків перевірки, майже кожна окрема частина буде перетворена в код Java, таким чином вона записується у файл Java як вихід.

Для того, щоб розібратися з особливостями цього алгоритму перетворення коду, необхідно порівняти мови програмування і визначити відмінності синтаксису однієї від іншої. Саме відмінності та особливості синтаксису впливають на логіку конверторів коду і алгоритми перетворення мов програмування.

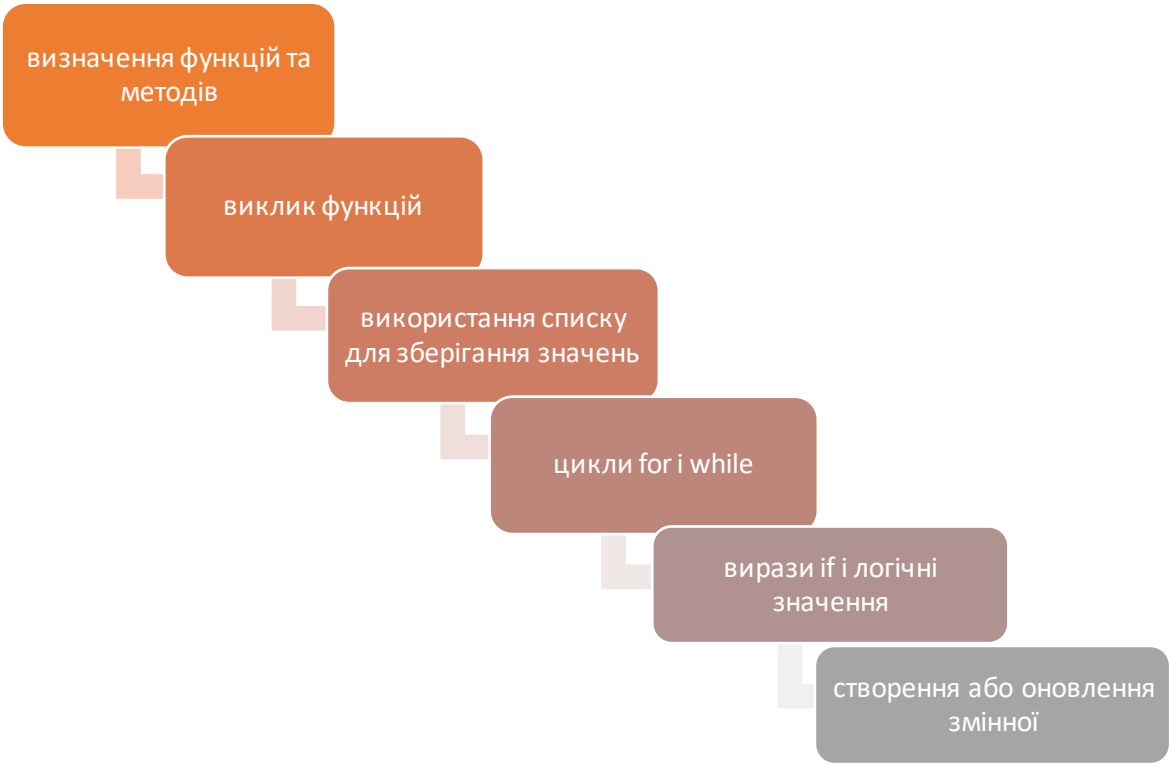

Рисунок 6 – Порядок перетворення окремих елементів коду

На перший погляд загальний алгоритм виглядає доволі простим і зрозумілим. Перетворенню підлягають синтаксичні конструкції, оператори, функції та змінні. Але саме у відмінностях типів і підходів до створення кодів програм і полягають труднощі конвертування між цими мовами.

3.1. Відмінності синтаксису мов програмування

Коротко опишемо відмінності синтаксису цих двох мов. Python – це динамічно типізована мова програмування, яка була розроблена для виконання загальних цілей і була легко читабельною. Динамічно типізована означає, що Python виконує перевірку типу під час виконання. Його простота у використанні робить Python однією з найпопулярніших мов програмування, тому її часто порівнюють з Perl, Ruby і, звичайно, Java. Загалом Python

вважається дуже потужною та зрозумілою об’єктно-орієнтованою мовою програмування. Він сумісний з усіма операційними системами (Windows, macOS, Unix і Linux). Крім того, Python є безкоштовним програмним забезпеченням. Це означає, що користувачі можуть завантажувати та використовувати Python безкоштовно, а мову також можна вільно поширювати або змінювати, оскільки вона доступна під GNU General Public License (GPL).

Java – це також об’єктно-орієнтована мова програмування загального призначення зі статичною типізацією (виконує перевірку типу під час компіляції). Java скомпільовано в байт-код, який можна запускати на будь-якій віртуальній машині Java (JVM). Таким чином, ця мова не залежить від платформи, оскільки її не потрібно компілювати на певній платформній машині. Синтаксис Java схожий на C++ і C; однак він має менше об’єктів нижчого рівня.

Автори статті [7] зробили спробу відобразити відмінності у вигляді таблиці (табл. 1), яка наочно відображає на що треба звернути увагу у процесі розробки алгоритму перетворення.

Таблиця 1

Порівняння мов програмування Python і Java

Властивість	Python	Java
Типи мови	Інтерпретована мова, яка може миттєво перетворювати зрозумілий людині код на машинний	Компільована та інтерпретована мова, її вихідний код компілюється у двійковий байт-код, який, у свою чергу, працює на JVM
Популярність серед розробників	Трохи популярніший на даний момент Python має невелику перевагу в популярності над Java (рис. 7, 8, 9)	До 2021 року Java випереджала Python за популярністю, але зараз має менше користувачів, ніж Python

Продовження табл. 1

Властивість	Python	Java
Типи даних	Python є інтерпретованою мовою, динамічно типізований	Java є скомпільованою та інтерпретованою мовою, яка типізується статично
Види додатків	Наукові та численні обчислення; програми машинного навчання; обробка зображення; розвиток мови	Веб-додатки; настільні програми GUI; корпоративні рішення; вбудовані системи
Особливості синтаксису	Менше коду – розробнику не потрібно вводити змінні, оскільки вони вводяться під час виконання; не потребує фігурних дужок або правил позначення	Більше коду – розробник повинен вводити всі змінні та має дуже суворі правила синтаксису
Продуктивність	Компілює код під час виконання, тому не є таким гнучким щодо компіляції з кожною платформою	Java компілює код заздалегідь і розповсюджує байт-код. Завдяки синтаксису статичного введення Java компіляція насправді швидша та легша, ніж динамічний набір у Python
Стабільність	Python перевіряє синтаксис під час виконання, тому не є таким стабільним, як Java, незважаючи на швидкий розвиток	Вища стабільність. Оскільки перед запуском все потрібно перевірити та встановити, код має бути дуже добре написаний, а отже, програмне забезпечення може бути стабільнішим і менш схильним до збоїв

Продовження табл. 1

Властивість	Python	Java
Швидкість розробки	Швидкий розвиток завдяки легкості, простоті та практичності написання на Python	Java-проекти, як правило, займають більше часу і можуть потребувати більшої групи розробників
Навчання	Легше у вивченні, підходить для початківців, оскільки його синтаксис досить простий і зрозумілий	Навчання займає більше часу, існує крива навчання з високою початковою точкою

Python і Java є двома потужними конкурентами, які борються за лідерство серед мов програмування. Обидві мови виділяються своїми можливостями для виконання більшості завдань з інформатики. Таким чином, завдання вибору мови для написання коду є складним і потребує ретельного аналізу. Згідно зі щорічним опитуванням Stackoverflow, JavaScript все ще залишається найпопулярнішою мовою восьмий рік поспіль. Однак Python (44,1%) і Java (40,2%) мають дуже близький відсоток користувачів.

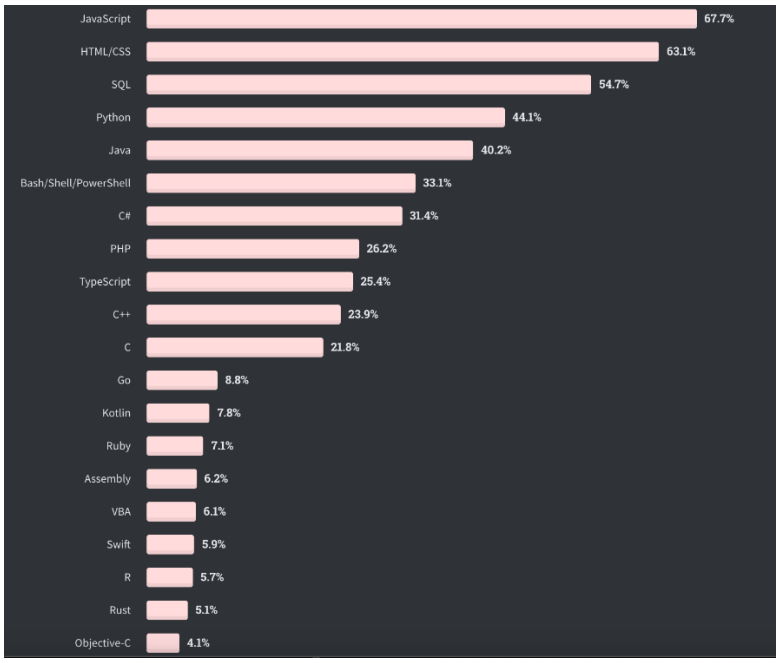

Рисунок 7 – Популярність мов програмування (джерело: stackoverflow.com)

Подібні результати представлені Octoverse GitHub. Графік показує, що між 2018 і 2019 роками Java втратила невелику перевагу над Python. Однак це не означає, що конкуренція між обома мовами закінчилася, оскільки (на сьогоднішній день) Python ще не зміг зберегти значну перевагу.

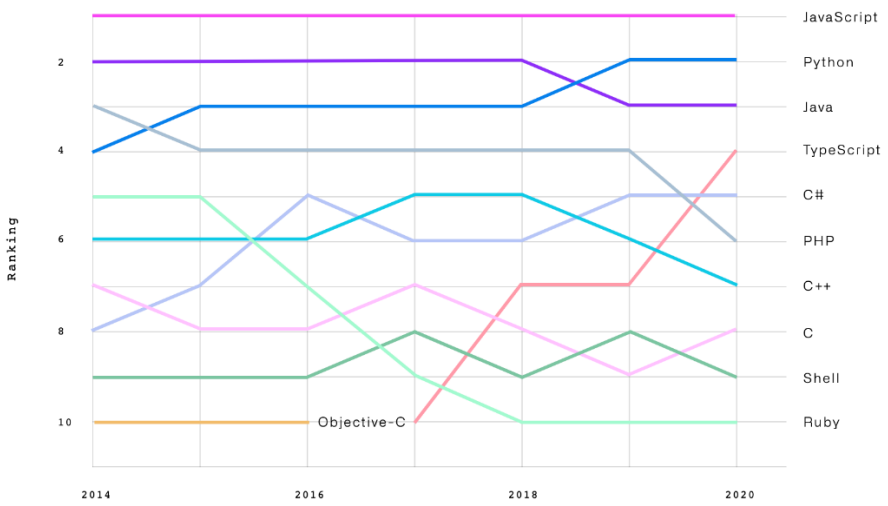

Рисунок 8 – Популярність мов програмування
(джерело: GitHub Octoverse)

Дані Google Trends свідчать, що в усьому світі спостреігається зростання популярності Python над Java протягом останніх років. Однак це збільшення відбулося зовсім недавно, враховуючи, що до 2015 року Java мала явну перевагу.

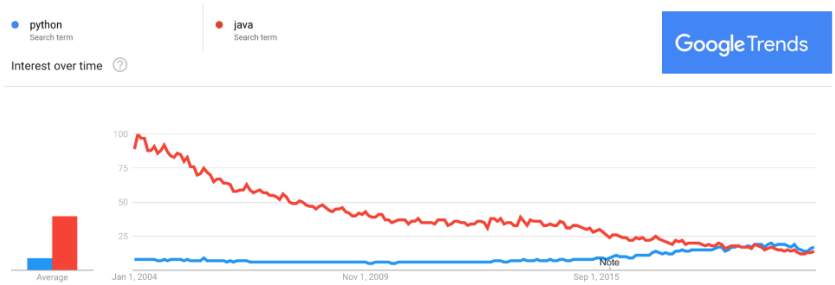

Рисунок 9 – Популярність мов програмування
(джерело: Google Trends)

З інформації, яка наведена у таблиці 1, визначаємо, що мови суттєво відрізняються одна від іншої. Але для алгоритму перетворення коду основне значення мають відмінності синтаксису двох мов. Розглянемо ці відмінності з прикладами. Як згадувалося раніше, Python – це мова з динамічною типізацією, а Java – зі статичною типізацією. Це найбільш суттєва відмінність між цими об'єктно-орієнтованими мовами, оскільки вона впливає на те, як розробник пише, проектує та фіксує помилки.

Під час введення коду розробник не вимагає введення змінних, оскільки вони вводяться під час виконання, що робить Python дуже легкою та зрозумілою мовою. Це робить Python дуже зрозумілим і легким для читання. Ще однією перевагою є те, що ця мова не потребує фігурних дужок, а блоки коду організовані відповідно до відступів, що робить Python дуже зручним та інтуїтивно зрозумілим. Наприклад, для оголошення змінних достатньо написати наступне:

```
# оголошення змінних
x, y = 12, 10
isTrue = True
greeting = "Welcome!"
```

На відміну від Python Java вимагає від розробника введення всіх змінних і має дуже суворі правила синтаксису:

```
public class Main {
    public static void main(String[] args) {
        // оголошення змінних
        int x = 12, y = 10;
        boolean isTrue = true;
        String greeting = "Welcome!";
    }
}
```

Щоразу, коли в коді є помилка, програма не запускається, що може бути складним, особливо для новачків. Наприклад, Java потребує десять рядків коду для читання з файлу; Python, з іншого боку, вимагає лише двох рядків коду.

На відміну від Python (який допускає відступи для запису блоків із кількома рядками), Java потребує вставлення рядків у фігурні дужки, щоб установити та визначити метод або блок.

Наступні приклади порівняння Python із Java демонструють відмінності щодо подібних функцій:

1. Списки та масиви

Python:

```
# робота з масивами даних
countries = [
    "Portugal",
    "England",
    "Brazil",
    "New Zealand",
    "Spain"
    ]
numbers = [12, 14, 9, 10, 9]
# сортування масиву
countries.sort()
# робота з елементами масиву у циклі
for country in countries:
    print(country)
```

Java:

```
import java.util.Arrays;
public class Main {
    public static void main(String[] args) {
        // робота з масивом
```

```
String[] countries = {
    "Portugal",
    "England",
    "Brazil",
    "New Zealand",
    "Spain"
};
int[] numbers = {12, 14, 9, 10, 9};
// сортування масиву у порядку зростання
// вимагає import java.util.Arrays
Arrays.sort(countries);
// робота з елементами масиву у циклі
for (String city : countries) {
    System.out.println(city);
}
```

Приклади демонструють відмінності роботи з масивами, а саме, оголошення масиву та його ініціалізація, відмінності функцій сортування та інших функцій роботи з елементами масиву. Крім цього, для послідовного перебору елементів необхідні циклі, вони теж відрізняються за синтаксисом.

2. Визначення класу з конструктором і методом

Python:

```
# Defining a class with constructor and a method
class Person:
    def __init__(self, name, nationality, age):
        self.name = name
        self.nationality = nationality
        self.age = age
    # Declaring a method with if/else statement
```

```
    # and returning a boolean
    def isMinor(self):
        if self.age >= 18:
            return False
        else:
            return True
```

Java:

```
// Defining a class with constructor and a method
public class Person {
    String name;
    String nation;
    int age;
    public Person(String name, String nation, int age) {
        this.name = name;
        this.nation = nation;
        this.age = age;
    }
    public boolean isMinor() {
        if (this.age >= 18) {
            return false;
        }
        else {
            return true;
        }
    }
}
```

Відмінність у роботі з конструкторами та методами також обумовлена типізацією змінних, тому одна мова вимагає оголошення змінних, а інші – ні. Це також має обмеження в процесі визначення методів класу.

3. Створення екземпляра об’єкта з класу person

Python:

```
# Instantiate an Object from class Person
p1 = Person("John Doe", countries[0], 19)
print(p1.isMinor())
```

Java:

```
// Instantiate an Object from class Person
 Person p1 = new Person("John Doe", countries[0], 19);
 System.out.println(p1.isMinor());
```

3.2. Обмеження алгоритму

Аналіз відмінностей синтаксису показав, що для створення алгоритму існує ряд обмежень, основні з яких викладені в роботі [5], і збігаються з результатами, які отримані автором. Перша фундаментальна відмінність між мовами Java і Python полягає в тому, що Python має структуру даних під назвою список, яка може зберігати будь-які елементи будь-якої довжини в одній структурі списку [8]. Простими словами, тип і довжину списку може змінюватися. Однак ця функція забезпечує зручність за рахунок використання додаткового простору пам’яті та уповільнення швидкості коду. При цьому Java має дві подібні структури: масив і ArrayList. Масив Java має подібний синтаксис до списку Python (наприклад, list[0]) – це код, який отримує перший елемент як у списку Python, так і в масиві Java. Однак масив має бути оголошений із фіксованою довжиною та фіксованим типом, що не поєднується з гнучкими функціями списку Python. З іншого боку, ArrayList потребує фіксованого типу для елемента, який він містить, але має змінну довжину списку, при цьому його синтаксис дуже відрізняється (рис. 10).

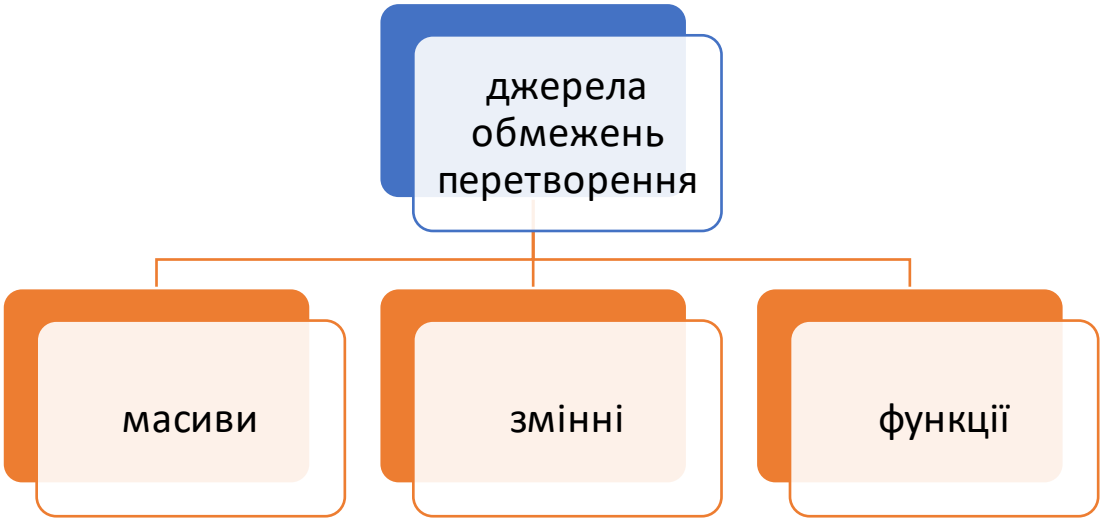

Рисунок 10 – Джерела обмежень алгоритму перетворення коду

Автори роботи [5] використовують ArrayList у вихідному коді Java, щоб замінити кожен список, створений у вхідному коді Python. Отже, (list = [ ]) перетворюється на ArrayList list = new ArrayList<>();. Щоб розв'язати фіксований тип даних, вони всі оголошуються як об'єкти, які є основним типом даних у Java. Однак це викликає складнішу проблему: приведення елементів, збережених як об'єктів у ArrayLists, до їх призначеного типу даних щоразу, коли вони використовуються.

Друге обмеження пов'язане з тим, що у Python можна оголосити змінну як ціле число, але пізніше змінити її значення на рядок. Синтаксис Python зазвичай не враховує типи даних і не генрує помилку компіляції для будь-яких помилок типу даних, що не відповідають оголошеному. Однак синтаксис Java суворий щодо типів даних. Після оголошення нової змінної їй потрібен фіксований тип даних. Таким чином, у вхідному рядку Python немає корисної інформації для типу змінної, окрім значення самої змінної. Алгоритм відстежує кожну створену змінну, як ім'я змінної, так і її значення, виявляючи рядки кодів, які містять ізольований один «=». Якщо ім'я змінної не є записом,

запускається серія складних кейсів для класифікації значення змінної та надання відповідним змінним Java належного типу. Це також включає окремий випадок ініціалізації ArrayList. За допомогою цього методу алгоритм може привести елементи з об’єктами типу в ArrayListonce, до якого вони мають отримати доступ.

Третє обмеження обумовлено тим, що кожна мова має власні бібліотеки та функції, які постійно розвиваються. При створенні конвертора це обов’язково слід враховувати. Алгоритм перетворення вирішує це за рахунок пошуку відповідної пари функцій. Наприклад, обидві мови містять математичну бібліотеку та можуть викликати функції для виконання математичних операцій. Функція Python pow(a,b) відповідає функції Java Math.pow(a,b). Тому необхідно створити словник відповідних функцій. Після визначення типової структури виклику функцій у вхідному коді Python словник повертає відповідну функцію Java. Використання такого словника дозволяє постійно поповнювати та удосконалювати алгоритм.

3.3. Алгоритм перетворення коду

Конвертер – це алгоритм, закодований на Python, який перетворює вихідний код Python у вихідний код Java за допомогою тієї ж функції. Він робить це в процесі рядок за рядком, подібно до методу перерахування. Зазвичай кожен рядок аналізується окремо, тобто алгоритми розглядають кожен рядок як такий, що перетворює те саме, ніби це єдиний вписаний рядок Python. Зосереджуючись на фактичному процесі, він використовує аналіз рядків для визначення певних структур у коді та перетворення їх у код Java. Структура, яку він шукає, є наступною в такому порядку: виклик або створення методів/функцій; умовний; приведення змінних; цикл for і while; робота зі змінними; список Python; та інші граничні випадки часто використовуваних функцій. Окрім построкового процесу, існує певна

інформація, яка є значущою для всього коду, наприклад змінні та методи, які використовуються в коді. Алгоритм спостерігає за змінними та функціями під час їх першого створення та зберігає їх разом із параметрами та типом для подальшого використання. Читання одного рядка Python, запис одного рядка Java після перетворення всього коду Python, код Java виводиться як файл Java [5].

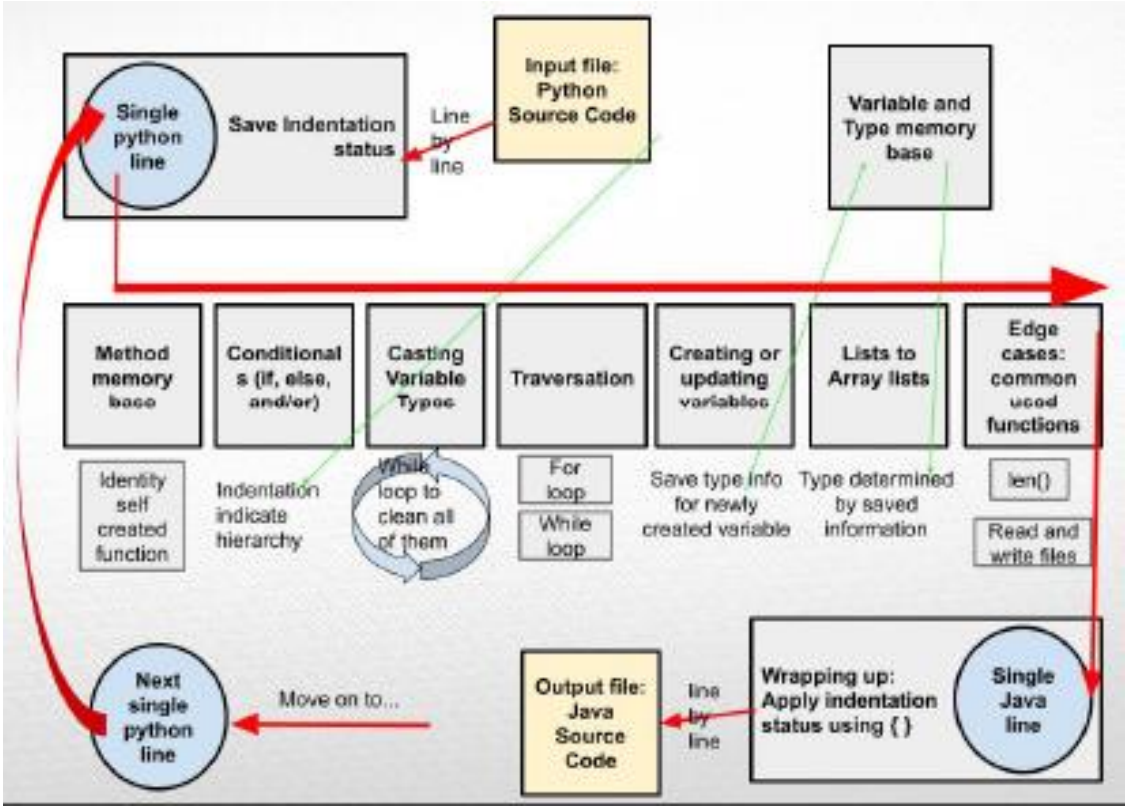

Рисунок 11 – Алгоритм перетворення коду

4. ПРОГРАМНИЙ МОДУЛЬ ПЕРЕТВОРЕННЯ КОДУ

4.1. Перетворення відступів, умовних конструкцій та функцій

Основою цього алгоритму є цикл for, який рядок за рядком читає файл, що містить вихідний код Python. створюється змінна «JavaLine», яка зберігає версію Java поточного рядка. Спочатку виявляється порожній рядок, тому код може продовжувати підраховувати кількість пробілів перед першим символом у цьому рядку. Це число, що вказує на відступ, записується в список, потім усі пробіли видаляються з рядка, тому лише значущий код передається на наступний аналіз. Цей розділ коду зосереджується на важливій інформації, яку відкриває відступ. Оскільки відступ Python визначає ієрархію коду, його потрібно перетворити на пари “{ “ і “}”, відповідну структуру в Java. Відкриття “{" буде подбати про пізніше в алгоритмі, коли “:” видно в кінці коду, тому в поточному розділі потрібно лише додати закриття “}” за потреби. Для цього алгоритм проходить список пробілів, коли рядок має менше місця, ніж попередній, це означає закриття попереднього рівня. Таким чином, буде додано кінцевий символ «}». Однак певні структури відступів неможливо визначити за допомогою цього методу, тому іноді користувачеві потрібно вручну виправити «{» і «}» у коді Java [5]. Це є недоліком вказаного алгоритму, але не знижує його цінність в реалізації інших завдань перетворення коду між мовами програмування.

```
list = []
#    a for loop adding value to list
for i in range(5):
list.append(i)
print(list)
```

```
ArrayList list = new ArrayList<>();
//    a for loop adding value to list
for(int i = 0; i < 5; i+=1 ) {
list.add(i);
}
 System.out.println(list);
```

Для обробки оператора if алгоритм вибирає частину коду, де Java і Python відрізняються, і перетворює її на версію Java. Цей специфічний сегмент простий, що робить його типовим прикладом. По-перше, якщо «if», «else» або «else if» видно на початку рядка, це негайно робить цей рядок частиною структури if-else без винятку. Таким чином, єдине, що потрібно зробити, це трохи відредагувати синтаксис, додавши дужки та дужки. Ще одна відмінність між Python і Java полягає в тому, як вони пишуть «true», «false» «if» «else», цикли потрібні, щоб знайти їх усі, оскільки в одному рядку може бути написано декілька таких конструкцій.

```
if a == 2 and (True or False):
        print("case 1")
else:
        print("case 2")
```

```
if( a == 2 && (true || false)) {
        System.out.println("case 1");
}
else {
        System.out.println("case 2");
}
```

Перший рядок коду визначає, чи є крапка. Змінна line є рядком, що містить рядок Python. У другому рядку перевіряється наявність крапки як частини рядка шляхом перевірки наявності лапок у рядку. Однак, навіть якщо його немає в рядку, інша небажана ситуація полягає в тому, що крапка використовується як десяткова крапка між числами, наприклад, 1,25 тощо, умова алгоритму виключення її – перевірка чи є символ перед крапкою числом. Якщо це не так, алгоритм отримав те, що шукав: виклик функцій. Зазвичай крапка в синтаксисі Python означає використання виклику функції з імпортованої залежності, яка зазвичай має структуру «caller.function (parameter)». Алгоритм використовує деяке нарізання рядків для відокремлення кожної частини, серед цих частин функція замінюється версією функції Java шляхом виклику словника func(), що містить цю інформацію. Цей параметр також важливий, коли параметр є списком, Java потребує створення

нового ArrayList, саме це і робить останній кусок коду у прикладі. Після отримання всіх частин у Java алгоритм знову збирає їх в один рядок Java, викликаючи функцію Java.

Це частина коду бази пам'яті функцій, яка використовується для збереження взаємозамінних функцій Python і Java. З часом його можна оновити, щоб включити більше пар, залежно від залежностей, які використовуються в коді Python.

list1.append([ ])	*ArrayListnewArray = new ArrayList<>();*
list2.pop(0)	*list1.add(newArray); list2.remove(0)*

Для встановлення відповідності між функціями створено базу функцій, яка використовується для збереження взаємозамінних функцій Python і Java:

```
def func(function, para):
    javafunc["append"] = "add"
    javafunc["readline"] = "readLine"
    javafunc["read"] = "readLine"
    javafunc["write"] = "println"
    javafunc["close"] = "close"
    javafunc["remove"] = "remove"
    javafunc["index"] = "indexOf"
    javafunc["pop"] = "remove"

return javafunc[function]
```

Вона створена таким чином, щоб мати можливість поповнювати її згодом, це обумовлено видами залежностей, які використовуються в коді Python. Це є додатковою перевагою конвертора тому, що дозволяє реагувати на особливості коду, який необхідно перетворити. Зміни в такому списку дозволяють налаштувати конвертор під конкретні задачі перетворення. Це є перевагою використання саме цього алгоритму.

4.2. Приведення типів змінних, циклу з параметрами та циклу з умовами

Частини коду, які перетворюють цілі числа, наприклад, int(a) на (int) або перетворюють інші типи змінних (рядок, логічний, float тощо) мають однакову логіку, тому розглянемо процес розпізнавання типів та їхнього перетворення. Розпізнавання приведення змінних відрізняється від інших структур, оскільки воно може з'являтися багато разів в одному рядку Python, що робить цикл while придатним інструментом для пошуку та зміни всіх змінних.

Це також означає, що алгоритму потрібно буде ідентифікувати пару дужок, що належать до цієї конкретної дії приведення. Після ідентифікації будь-якого «int(», який все ще існує в поточному рядку, алгоритм починає підраховувати дужки після нього. Тільки якщо лічильник (змінна дужкок) помічає, що кількість відкритих і закритих дужок збігається, можна визначити, яка частина коду є фактичним приведенням. Таким чином, застосовано той самий процес розбиття рядка та нагадування його синтаксису Java, і це конкретне приведення було успішно перетворено. Однак алгоритм не рухатиметься, доки не буде виконано все приведення поточного рядку, що робить його здатним обробляти вкладені операції.

int(number1)	*(int) number1;*
float(number2)	*(double) number2;*
str(x)	*String.valueOf(x);*
float (str (int (a)))	*(double) String.valueOf((int) a) ;*

```
while “int(“ in line:
    exist = False
    parentheses = 0
    inside = “”
    for I in range(3,len(line)):
```

```
if not exist and line[i] == “(”:
        if not exist and line[1-len(“int”): 1] == “int”:
        front = line[:i-len(“int”)]
        exist = True
        parentheses=1
        continue
if exist:
        if line[i] == “(”:
        parentheses+=1
        if line[i] == “)”:
        parentheses-=1
if parentheses == 0:
        line = front + “(int)” + inside + line[i+1:]
        break
inside+=line[i]
```

Якщо рядок Python починається з «for», це точно цикл for (або цикл for each). Цикл Python for складається з 3 частин: початок, кінець, крок у формі «for i» в діапазоні (початок, кінець, крок). Версія Java буде для (int i = початок; i< кінець; i+= крок). Однак зазвичай єдиним параметром, який використовують програмісти, є лише кінець, щоб повторити цикл x разів. Отже, алгоритм розпізнає різні ситуації, розділяючи рядок Python і підраховуючи кількість ком у рядку діапазону (). Якщо крок або початок не використовуються, для них буде встановлено значення за замовчуванням 1 і 0. Тоді всі частини, включаючи змінну (var), початок, кінець і крок, схожі на синтаксис Java. Однак нарізка рядків має недоліки, пов’язані з нездатністю мати справу із зайвими пробілами в рядку Python, тому вона залежить від припущення синтаксису вписаного коду Python. Подібним чином, цикл for each перетворюється шляхом захоплення окремого елемента та групи елементів із рядка Python і розміщення їх у форматі Java «for (Object element: elements) {“. Важливо запам’ятати, що типом встановлено Object, оскільки його неможливо визначити з вихідного коду Python.

for i in range(10):	*for(int i = 0; i < 10; i+=1) {*
for a in range(0,50,2):	*for(int a = 0; a < 50; a+=2) {*
for b in range(n):	*for(int b = 0; b < n; b+=1) {*
for value in elements:	*for(Object value : elements)*

З іншого боку, цикл while набагато простіший за цикл for. Просто виявлення рядка, що починається з «while», усуне інші можливості:

while a== 2:	*while (a == 2) {*

Процес перетворення полягає у виділенні логічного оператора та розміщенні його в дужках:

```
if line[0:6] == “while”:
    condition = line[6:len(line)-1].strip()
    javaLine = “while ( “+condition+” ) {”
    print(javaline)
    java.write(javaLine + “\n”)
    print(“”)
    continue
```

Алгоритм визначає знаки рівності «=» у поточному рядку. Якщо перед знаком рівності стоїть знак математичної операції, це означає, що версія Java збігається з версією Python. Таким чином, рядок просто записується у файл Java, оскільки на цьому етапі алгоритму інші частини Python у поточному рядку вже перетворені. Однак у випадку ізольованого знака рівності створюється змінна або змінюється значення змінної. Виникає головна відмінність між Python і Java: Java вимагає призначення фіксованого типу змінним, які створюються першими. Алгоритм протидіє цій проблемі, зберігаючи створені змінні до словника (varName : type), тому оголошені змінні можуть бути вилучені з новостворених змінних. Рядок із створенням змінної містить два компоненти: ім’я змінної зліва та будь-яке значення з правого боку. Виявивши певні особливості значення, наприклад, лапки для

рядка та «[ ]» для ArrayList, у більшості випадків можна визначити фіксований тип для них. Потім ця інформація зберігається в словнику, який було описано раніше. Однак через велику різницю в тому, що Python розглядає все як об’єкт, у деяких випадках тип змінної неможливо визначити, просто дивлячись на код Python, тому користувачеві потрібно вручну надати цю інформацію.

```
a = 2                          int a = 2;
a = a * 3                      a = a * 3;
name = “Eric”                  String name = “Eric”;
name = name + “Jin”            name = name + “Jin”;
list = []                      ArrayList list = new ArrayList<>();
```

ВИСНОВКИ

В роботі була проаналізована предметна область, обрані інструменти для розробки та розроблено алгоритм перетворення програмного коду. Далі на основі цього алгоритму було розроблено елементи конвертора програмного коду на мові програмування Python, який перетворює код, написаний на Java.

На першому етапі аналізу предметної області визначили функціональні можливості та напрямки використання конверторів програмного коду в процесах розробки програмного забезпечення для різноманітних задач. За результатами аналізу обрали програмні інструменти для створення конвертора коду. Проаналізували обмеження перетворення програмного коду з метою удосконалення алгоритму. Для визначення вимог до проєкту та вибору мов програмування проаналізували існуючі транслятори коду, їх функціональні можливості, виділили переваги та недоліки. В результаті проведеного дослідження визначили, що конвертор програмного коду буде перетворювати код Java у код Python.

На другому етапі представили алгоритм перетворення програмного коду, який враховує відмінності синтаксису обраних мов програмування. Найбільш суттєва відмінність між цими об'єктно-орієнтованими мовами полягає у способі типізації. Python – це мова з динамічною типізацією, а Java – зі статичною типізацією. Визначили три основних джерела обмежень алгоритму: масиви, змінні, функції. Для останніх створили спеціальний словник відповідностей, який постійно поповнюється.

На заключному етапі було представлено алгоритми і рішення для перетворення основних конструкцій: відступи, приведення змінних, умовні оператори, цикли та виклик функції. Основою цього алгоритму є цикл з параметрами, який рядок за рядком читає файл з програмним кодом Python, створює спеціальну змінну, яка зберігає версію Java поточного рядка. Недоліком розробленого алгоритму є необхідність перевірки вихідного коду

після перетворення та поповнення словника функцій в залежності від особливостей вихідного коду.

Подальші дослідження можуть бути пов’язані з удосконаленням алгоритму перетворення програмного коду, розробкою інструменту для встановлення відповідностей між вбудованими функціями та автоматичного поповнення словника функцій.

ПЕРЕЛІК ПОСИЛАНЬ

1. Open Source Code Converters / https://sourceforge.net/directory/code-converters / (дата звернення 10.04.2023).

2. 6 Tools To Translate Code Between Programming Languages, Nivash Jeevanandam / https://analyticsindiamag.com/6-transcompiler-tools / (дата звернення 10.04.2023).

3. The value and limitations of mainframe code conversion/ https://planetmainframe.com/2022/11/the-value-and-limitations-of-mainframe-code-conversion / (дата звернення 10.04.2023).

4. Tools to translate coding languages / https://analyticsdrift.com/tools-to-translate-coding-languages/(дата звернення 30.04.2023).

5. Eric Jin, Yu Sun. An algorithm-adaptive source code converter to automate the translation from Python to Java / https://aircconline.com/csit/ papers/vol11/csit111719.pdf / (дата звернення 30.04.2023).

6. Beckwith M., Frank R. Process of enumeration. Psychological Review 73.5. 1966. – 437 p.

7. Python vs Java: key differences and code examples / https://www. imaginarycloud.com/ blog/python-vs-java/ (дата звернення 30.04.2023).

8. Arnold, Ken, James Gosling, and David Holmes. The Java programming language. Addison Wesley Professional, 2005. – 436 p.

9. Qiu, Lili. Programming language translation. Cornell University, 1999. – 346 p.