УДК 004.5
DOI https://doi.org/10.32782/2663-5941/2024.2/12

Дьячук Т.С.
Національний університет «Запорізька політехніка»

Шкрябець В.І.
Національний університет «Запорізька політехніка»

Тіменко А.В.
Національний університет «Запорізька політехніка»

Голуб Т.В.
Національний університет «Запорізька політехніка»

АВТОМАТИЗОВАНА СИСТЕМА ГЕНЕРАЦІЇ ЗАВДАНЬ В НАВЧАЛЬНИХ КУРСАХ З ПРОГРАМУВАННЯ

Наразі дистанційна освіта стала практично єдиною можливістю отримати знання, спочатку через пандемію COVID-19, а згодом через повномасштабну війну в Україні, особливо у прифронтових містах. Стаття присвячена вирішенню практичної задачі розробки автоматизованої системи генерації завдань студентів при навчанні програмуванню, яка допоможе викладачу організувати ефективний та якісний навчальний процес в умовах дистанційної освіти, часто асинхронний через воєнний стан. Система дозволяє надати студентам різноманітні завдання, заощадити час викладача та отримати більш об'єктивну оцінку знань студентів, оскільки кожен отримує унікальне завдання. Галузь використання системи – університети, заклади освіти та компанії, що мають курси з навчання програмуванню. Розроблювана система є частиною більш складної системи для дистанційної освіти, тому згенеровані завдання придатні для подальшої автоматизованої перевірки та мають чітко визначену специфікацію. Для досягнення поставленої мети було: проведено аналіз публікацій за темою автоматизованого навчання, спроєктовано, розроблено та реалізовано автоматизовану систему генерації завдань, впроваджено її у навчальний процес в рамках дисципліни «Основи програмування на Kotlin». Розроблена система охоплює різні теми навчання. Спочатку викладачу потрібно зробити базове завдання, та певний набір підзавдань, з яких буде формуватися варіативність результуючого завдання, для кожної теми навчання. Отримані результуючі завдання залежать від ідентифікатора студента, для перетворення якого використовується хеш-функція, що гарантує унікальність результуючого варіанту завдання з кожної теми. Для реалізації системи використані: мова Kotlin для розробки системи, система автоматизації збірки Gradle, середа розробки InteliJ Idea, система керування версіями файлів Git, GitHub – репозиторій для вихідного коду. Система генерації в скомпільованому виді викладена у GitHub репозиторій та доступна студентам у формі бібліотеки, яку можна підключати до своїх проєктів та використовувати при вивченні навчального матеріалу. Бібліотека зроблена у вигляді Maven артефакту. Робота над системою продовжується, репозиторій постійно оновлюється, виправляються помилки, та система поліпшується.

Ключові слова: *GitHub, Kotlin, LaTeX, MD5, Gradle, Maven артефакт, хеш-функція, дистанційна освіта.*

Постановка проблеми. У сучасному світі вміння програмувати є дуже корисною навичкою, особливо для студентів технічних спеціальностей. Воно сприяє розумінню технологій, розвиває творчість, критичне та проблемно-орієнтоване мислення, відкриває нові можливості для майбутньої кар'єри.

На даний час дистанційна освіта стала практично єдиною можливістю отримати знання, спочатку через пандемію COVID-19, а згодом через повномасштабну війну в Україні, особливо у прифронтових містах. Але в цілому вона стає доволі популярною сама по собі у контексті сучасних технологій, та надає багато переваг для студентів у зручному та ефективному способі отримання освіти. Вона дозволяє студентам навчатися зручно та безпечно, обираючи режим, який підходить їм найкраще. Однією з ключових інновацій в цій сфері є автоматична генерація та перевірка завдань. У цій статті приділено увагу саме генера-

ції завдань, які в подальшому можливо перевіряти завдяки автоматизованої системи перевірки.

Процедурна генерація завдань – це процес автоматичного створення завдань або тестів з використанням алгоритмів та програмного забезпечення. Цей підхід використовується в багатьох галузях, включаючи освіту, наукові дослідження та інженерію програмного забезпечення. У навчальних курсах з програмування генерація завдань може бути корисною для автоматичного створення індивідуальних завдань на основі рівня знань кожного студента. Завдяки такому підходу, викладачі можуть надати студентам різноманітні завдання, заощадити час та зосередитися на навчанні і оцінюванні результатів. Це також дозволяє отримати більш об'єктивну оцінку знань, оскільки кожен отримує унікальне завдання. Однак, при процедурній генерації завдань необхідно враховувати різноманітні фактори, такі як рівень складності, обсяг та якість згенерованих завдань, а також їх придатність для оцінювання знань.

Аналіз останніх досліджень і публікацій. Останні дослідження та публікації за цією темою свідчать про високу актуальність та перспективність систем дистанційної освіти у тому числі при навчанні програмуванню. Наприклад, автори статті [1] прийшли до висновку, що хоча студенти і висловлювали негативне ставлення до автоматизованої системи, але автоматичне машинне оцінювання для навчання програмуванню краще готує студентів до ситуацій, коли їм доводиться писати код самостійно, усуваючи залежність від зовнішніх джерел допомоги та мотивуючи розвиток самодостатності. У роботі [2] автори розглядають можливість використовувати OpenAI Codex як велику мовну модель при створенні вправ з програмування (включаючи приклади розв'язків та тести) та поясненні коду для вступного навчання програмуванню. Дослідження вказують на значну цінність масштабних моделей машинного навчання як інструмента для викладачів, хоча і є потреба в контролі якості згенерованого контенту перед його передачею студентам. У статті [3] розроблена система 2TSW, яка спрямована на покращення навчання програмуванню шляхом автоматизованої корекції завдань із можливістю отримання зворотного зв'язку. Ця система використовує ігровий підхід, що дозволяє студентам отримувати бали досвіду та підвищує їхню мотивацію через змагальний елемент. Результати експерименту показали, що студенти високо оцінюють систему та підтримують високий рівень зацікавленості.

Отже, на даний момент існує безліч систем, розробок та підходів до автоматизованого навчання, тому ми вважаємо цю тему затребуваною, актуальною та такою, що потребує подальших досліджень.

Постановка завдання. Метою статті є розробка автоматизованої системи генерації завдань студентів при навчанні програмуванню, яка допоможе викладачу організувати ефективний та якісний навчальний процес в умовах дистанційної освіти, часто асинхронний через воєнний стан. Розроблювана система є частиною більш складної системи для дистанційної освіти, тому згенеровані завдання повинні відповідати певним критеріям та мати чітко визначену специфікацію для подальшої автоматизованої перевірки.

Виклад основного матеріалу. Розглянемо етапи, з яких складається наша система генерації завдань (рис. 1). Спочатку береться унікальний ідентифікатор студента – рядок символів (наприклад логін), що гарантує унікальність результуючого варіанту завдання. Далі хеш-функція перетворює цей ідентифікатор в дані фіксованого розміру – ланцюжок байтів. У якості алгоритму хешування було обрано алгоритм MD5. Хоча MD5 вважається застарілим і вразливим, для наших цілей він підходить, адже генерація завдань не потребує високого рівня рандомізації на відміну від задач пов'язаних з кібербезпекою, а у JVM MD5 можна використовувати для створення хеш-кодів за допомогою вбудованих бібліотек та класів.

Викладачу потрібно зробити базове завдання, та певний набір підзавдань, з яких буде формуватися варіативність результуючого завдання, для кожної теми навчання. Тему навчання задає ідентифікатор теми – натуральне число (порядковий номер теми). Ідентифікатор теми навчання визначає яке саме базове завдання буде обрано та принцип, за яким будуть формуватися підзавдання (який набір вправ буде використано та які байти з ланцюжку будуть впливати на вибір конкретних підзавдань з обраної теми).

Далі формується структура завдання з базового завдання та набору підзавдань, яка буде використовуватися в подальшому для автоматичної перевірки результатів роботи студента. А для зручності сприйняття структура завдання перетворюється у текстовий формат, який вже отримує студент для виконання. Таким чином, кожен студент отримує унікальне завдання.

Для прикладу розглянемо формування завдання з теми «Змінні та типи даних, рядки,

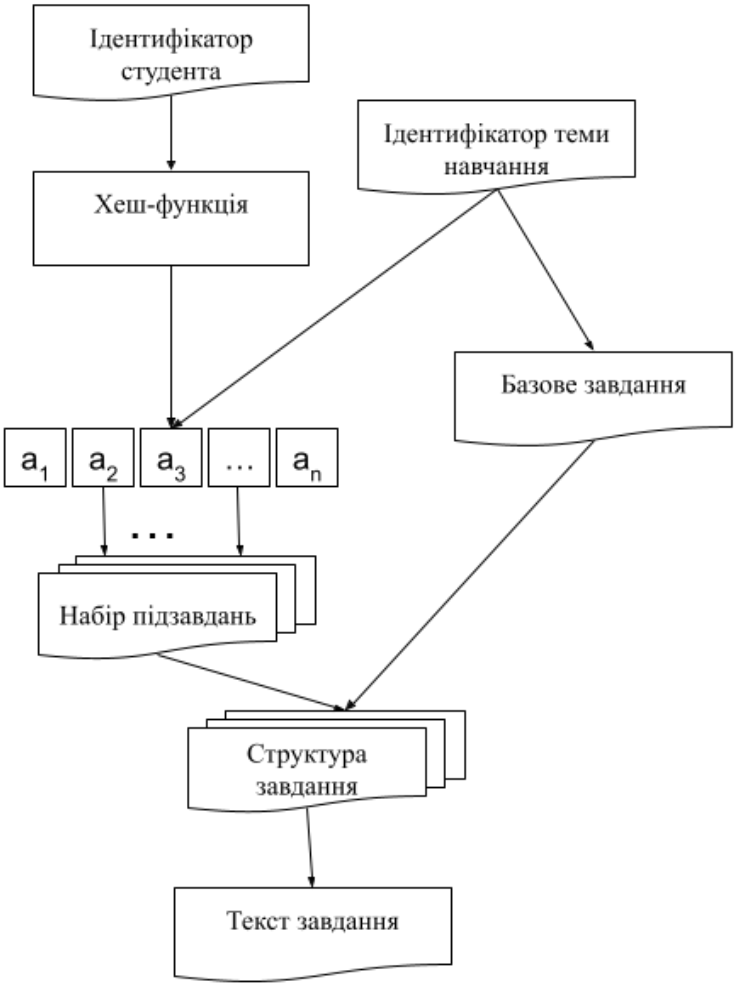

Рис. 1. Етапи генерації завдань

умови та цикли, функції» для дисципліни «Основи програмування на Kotlin». Зазначимо, що Kotlin [4] – сучасна статично типізована, об'єктно-орієнтована мова програмування, яка працює поверх JVM. Може застосовуватися для розробки застосунків у багатьох проблемних областях. Система генерації завдань також розроблена на мові Kotlin [4] з використанням середовища розробки IntelliJ IDEA [5].

Розглянемо приклад опису завдання у лістингу 1. Ця структура містить чотири поля даних: digest – масив байтів, отриманий хеш-перетворенням ідентифікатора студента. Від нього залежить варіативна частина, тому воно є обов'язковим для всіх завдань; intTask – завдання за темою «функції та операції з цілими типами даних»; doubleTask – завдання за темою «функції та операції з типами даних з плаваючою комою»; stringTask – завдання за темою «функції та операції з рядками».

Лістинг 1. Структура з описом загального завдання

```
1. data class TaskL2(
2.    val digest: ByteArray,
3.    val intTask : BaseTask<Int>,
4.    val doubleTask : BaseTask<Double>,
5.    val stringTask : StringTask,
6. ) : Task()
```

У лістингу 2 наведено обчислення значення digest за значенням ідентифікатора студента seedStr.

Лістинг 2. Обчислення значення digest

```
1. val seed = seedStr.toByteArray(Charset.
   forName("UTF-8"))
2. val md = MessageDigest.getInstance("MD5")
3. val digest = md.digest(seed)
```

В першому рядку ми перетворюємо рядковий ідентифікатор студента в масив байтів. Наступ-

ним кроком використовуємо бібліотечний клас MessageDigets та розраховуємо MD5 хеш від масиву байт.

На прикладі intTask (з лістнигу 1) розглянемо як відбувається формування завдання за значеннями з digest. В цьому завданні студенту потрібно написати функцію, яка має провести обчислення за певними вхідними даними та повернути результат обчислень. Загальна формула для такого завдання буде наступною:

$$res = mainFnc\left(secondaryFnc\left(x_0, x_1, \ldots\right)\right), \quad (1)$$

– mainFnc – головна функція над вхідними аргументами. Можливі варіанти: квадратний корінь (має ідентифікатор функції 0), кубічний корінь (ідентифікатор 1), косинус (ідентифікатор 2), тангенс (ідентифікатор 3), синус (ідентифікатор 4), тангенс гіперболічний (ідентифікатор 5), логарифм натуральний (ідентифікатор 6). Ідентифікатор функції обирається за формулою (min=0, max=6 – для цього випадку):

$$mainFncIndex = \left|digest_1\right| \% \left(max - min\right) + min; \quad (2)$$

– secondaryFnc – вторинна функція над вхідними аргументами. Можливі варіанти: сума (ідентифікатор 0), сума квадратів (ідентифікатор 1), сума кубів (ідентифікатор 2), мінімальне (ідентифікатор 3), максимальне (ідентифікатор 4), сума модулів (ідентифікатор 5), модуль мінімального (ідентифікатор 6), модуль максимального (ідентифікатор 7), множення (ідентифікатор 8), функція обирається за формулою (min=0, max=7 – для цього випадку):

$$secondFncIndex = \left|digest_2\right| \% \left(max - min\right) + min; \quad (3)$$

– $x_0, x_1, \ldots$ – вхідні аргументи.

Після розрахунку значення digest система формує структуру за кожною темою з описом окремого завдання. Для нашого приклада (intTask) структуру наведено в лістингу 3. В цьому завданні студенту потрібно написати функцію, яка має провести обчислення за певними вхідними даними та повернути результат.

Лістинг 3. Структура з описом окремого завдання

```
1. data class BaseTask<T>(
2.    val cnt: Int,
3.    val defValues: List<T>,
4.    val mainFnc: MainFunctions,
5.    val secondFnc: SubFunctions,
6. )
```

Отже структура окремого завдання містить наступні поля даних:

– cnt – кількість вхідних аргументів в межах від min=2 до max=5, розраховується за формулою (4):

$$cnt = \left|digest_0\right| \% \left(max - min\right) + min. \quad (4)$$

– defValues – вхідні аргументи (значення за замовченням для функції):

$$defValues_i = digest_i. \quad (5)$$

– mainFnc та secondFnc відповідають загальній формулі (1).

Наступним кроком буде створення тексту завдання. Воно формується з попередньої структури (лістинг 3) та має бути зрозумілим для сприйняття студентами. Чітко сформульовані завдання зменшують можливість непорозумінь та покращують якість виконаної роботи. Формули завдання відображуються в графічному вигляді, для цього блок генерації завдання використовує технологію LaTeX, яка надає потужні засоби для написання складних математичних формул, що дозволяє зробити їх зрозумілими та естетично оформленими у документах.

Для перетворення структури BaseTask в текстове завдання спочатку беремо кількість аргументів з структури BaseTask, первинну та вторинну функцію. Потім формуємо опис формули за стандартом LaTeX. І далі на етапі відображення графічного інтерфейсу перетворюємо LaTeX формулу в растрове зображення за допомогою бібліотеки jlatexmath [6].

На рис. 2 наведено текст сформованого завдання, яке отримує студент. Воно відповідає класу даних з лістинга 1. Завдання складається з трьох частин (intTask, doubleTask, stringTask) і відповідно має три вкладки. Для першої частини завдання (intTask) число змінних, значення за замовчанням та вигляд функції є варіативними та залежать від ідентифікатора студента. Базовою є умова створити функцію з певним типом та назвою. Друга та третя частини завдання (doubleTask, stringTask) також мають базові та варіативні умови.

Система генерації в скомпільованому виді викладена у GitHub репозиторій та доступна студентам у формі бібліотеки, яку можна підключати до своїх проєктів та використовувати при вивченні навчального матеріалу. Git та GitHub є дуже потужними інструментами для контролю версій та спільної роботи над проєктами. Бібліотека зроблена у вигляді Maven артефакту [7]. Для підключення бібліотеки в навчальних проектах потрібно у конфігураційний файл власного проєкту (build.

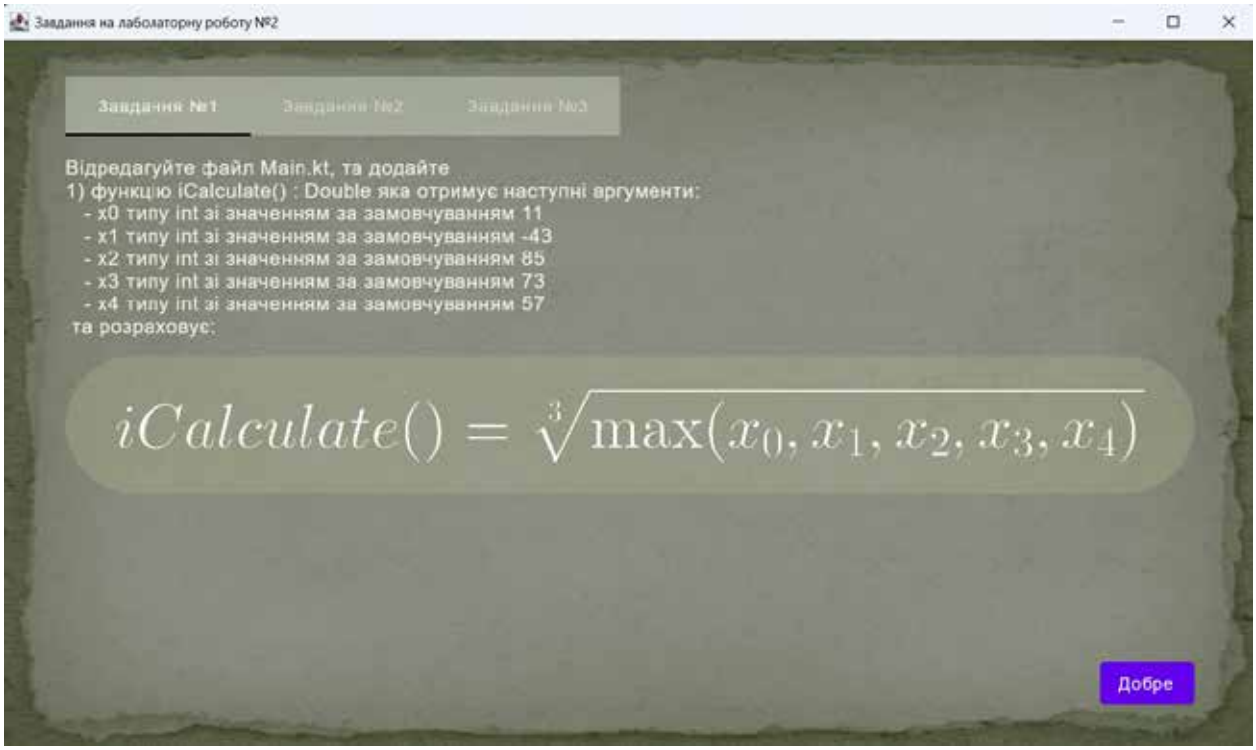

Рис. 2. Текст сформованого завдання

gradle) додати шлях до репозиторію з Maven артефактами, та увімкнути залежність від com.diacht.ktest:library бібліотеки (що представлено у лістингу 4, де "com.diacht.ktest:library:1.0.7" – поточна версія бібліотеки).

Лістинг 4. Підключення бібліотеки з системою генерації завдань у конфігураційний файл build.gradle

```
1. repositories {
2.    maven("file://${rootDir}/.m2repo/")
3.    …
4. }
5. dependencies {
6.    implementation("com.diacht.ktest:
      library:1.0.7")
7.    …
8. }
```

Висновки. Таким чином, автоматична генерація та перевірка завдань студентів у навчальних курсах з програмування може значно спростити процес викладання та оцінювання рівня знань студентів. Цей підхід дає змогу автоматизувати багато аспектів навчання та забезпечити об'єктивне оцінювання. Розроблена система дозволяє надати студентам різноманітні завдання, заощадити час викладача та отримати більш об'єктивну оцінку знань студентів, оскільки кожен отримує унікальне завдання. Галузь використання системи – університети, заклади освіти та компанії, що мають курси з навчання програмуванню.

Систему автоматизованої генерації завдань було розроблено, впроваджено та апробовано у рамках дисципліни «Основи програмування на Kotlin», вона охоплює різні теми навчання та має унікальні завдання з кожної теми, які залежать від унікального ідентифікатора студента. Розроблена система є частиною більш складної системи для дистанційної освіти, тому згенеровані завдання придатні для подальшої автоматизованої перевірки та мають чітко визначену специфікацію. Система в скомпільованому виді викладена у GitHub репозиторій та доступна студентам у формі окремої бібліотеки, що приєднується до системи збірки Gradle. Робота над системою продовжується, репозиторій постійно оновлюється, виправляються помилки, та система поліпшується.

Список літератури:

1. Maguire Ph., Maguire R., Kelly R. Using automatic machine assessment to teach computer programming, Computer Science Education, 2017, Vol. 27, Issue 3-4, P. 197-214, URL: https://doi.org/10.1080/08993408.2018.1435113 (date of access: 13.02.2024).

2. Sami Sarsa, Paul Denny, Arto Hellas, Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In Proceedings of the 2022 ACM Conference on International Computing Education Research V.1 (ICER 2022), August 7–11, 2022, Lugano and Virtual Event, Switzerland. ACM, New York, NY, USA, 17 pages. URL: https://doi.org/10.1145/3501385.3543957 (date of access: 14.02.2024).

3. Giuseppina Polito, Marco Temperini. A gamified web based system for computer programming learning, Computers and Education: Artificial Intelligence, 2021, Vol. 2, P. 100029, URL: https://doi.org/10.1016/j.caeai.2021.100029 (date of access: 27.02.2024).

4. Pierre-Yves Saumont. The Joy of Kotlin: Manning Publications, 2019 – 480p.

5. IntelliJ IDEA. URL: https://www.jetbrains.com/idea (date of access: 16.02.2024).

6. jlatexmath. URL: https://github.com/opencollab/jlatexmath (date of access: 26.02.2024).

7. DiachT/KotlinLabsNUZP – Maven репозиторій. URL: https://github.com/DiachT/KotlinLabsNUZP/tree/main/.m2repo/com/diacht/ktest/library (date of access: 01.03.2024).

Diachuk T.S., Shkriabets V.I., Timenko A.V., Holub T.V. AUTOMATED SYSTEM OF TASKS GENERATION FOR THE PROGRAMMING COURSES

Currently, distance education has become almost the only opportunity to gain knowledge, first due to the COVID-19 pandemic, and later due to the full-scale war in Ukraine, especially in the frontline cities. The article is devoted to the solution of the practical task of developing an automated system for generating tasks for students that are computer science. This system should help the teacher organize an effective and high-quality educational process in the conditions of distance education, often asynchronous due to the war. The system provides students with a variety of generated tasks, save the teacher's time and get a more objective assessment of students' knowledge, since everyone receives a unique task. The field of use of the system is universities, educational institutions and companies that have programming training courses. The developed system is part of a more complex system for distance education, so the generated tasks are suitable for further automated verification and have a well-defined specification. To achieve the goal, the following was done: an analysis of publications on the topic of automated learning was carried out, an automated task generation system was designed, developed and implemented, and it was implemented in the educational process within the framework of the discipline "Fundamentals of Kotlin Programming". The developed system covers various learning topics. First, the teacher needs to make a basic task and a certain set of subtasks, from which the variability of the resulting task will be formed, for each topic of study. The resulting tasks depend on the student ID, which is converted using a hash function, which guarantees the uniqueness of the resulting assignment option for each topic. The following tools are used to implement the system: Kotlin language for system development, Gradle build system, InteliJ Idea development environment, Git as version control system, and GitHub – repository for source code. The generation system in a compiled form is stored in the GitHub repository and is available to students in the form of a library that can be connected to their projects and used for studying educational material. The library is made in the form of a Maven artifact. Work on the system continues, the repository is constantly updated, errors are corrected, and the system is improved.

Key words: *GitHub, Kotlin, LaTeX, MD5, Gradle, Maven artifact, hash function, distance education.*