Санкт–Петербургский государственный университет
Факультет математики и компьютерных наук

Никита Алексеевич Босов

Выпускная квалификационная работа

Тема работы: Расширяемый генератор синтаксически корректных программ для обучения программированию

Уровень образования: бакалавриат
Направление 01.03.02 «Прикладная математика и информатика»
Основная образовательная программа СВ.5005.2018 «Прикладная математика, фундаментальная информатика и программирование»
Профиль «Современное программирование»

Научный руководитель:
д.ф.-м.н, профессор СПбГУ А.С. Куликов

Рецензент:
ассистент кафедры МОЭВМ ЛЭТИ
Н. В. Шевская

Санкт-Петербург
2022 г.

Содержание

Введение

В настоящее время знание языка программирования является необходимым для специалиста в отрасли информационных технологий, а обучение им - крайне востребованным. На сегодняшний день программы по обучению языкам программирования есть не только в университетах, но и на различных образовательных платформах в интернете. В связи с ростом числа учащихся подобных курсов и ослабления контакта между студентом и преподавателем острее встает проблема создания учебных материалов, в частности практических заданий. Требуется создавать их в большем объеме и в то же время делать их разнообразными во избежание списывания. Специфически для курсов по изучению языков программирования возникает необходимость создания множества примеров программ на определенную тему или по конкретному шаблону. Создание подобных примеров вручную в нескольких вариантах (в идеале по отдельности для каждого ученика) затруднительно. Таким образом, создание удобного программного инструмента, позволяющего автоматически генерировать примеры кода на различных языках программирования для учебных задач представляет собой актуальную проблему.

Постановка задачи

Целью данной выпускной работы является создать расширяемый генератор случайных программ для учебных задач, используемых в курсах по обучению языкам программирования.

Основные задачи которые необходимо сделать:

a. Изучить существующие системы генерации случайных программ на предмет возможности их настройки и применимости результатов их работы в учебных целях.

b. Создать систему генерации программ с возможностью настройки параметров для одного языка программирования (Python)

c. Адаптировать систему к возможности поддержки других языков программирования.

Объектом моего исследования являются инструменты генерации программного кода, а **предметом** исследования — применимость инструментов генерации кода для создания учебных задач.

Данная работа является развитием идеи, изложенной в статье [1], в сторону расширяемости и поддержки разных языков программирования.

Глава 1. Обзор и сравнение существующих генераторов программного кода

1.1 Понятие генерации программного кода

Генерация программного кода — это автоматическое создание программного кода специальным приложением, при котором по заданным условиям полностью или частично формируется исходный код программы. Такое специальное приложение называется **генератором кода**. Получается, что это программа, создающая программный код.

Основной сферой применения генераторов программного кода является автоматическое тестирования компиляторов. С помощью них можно обнаружить незаметные ошибки которые могут влиять на работу скомпилированного этими компиляторами программного обеспечения. В сравнение были включены несколько инструментов для тестирования компиляторов.

Также для поиска существующих аналогов был произведен поиск в поисковых системах “Google” и “Google Scholar” по следующим ключевым словам:

- “C++ program generator”
- “program generator”
- “random program generator”
- "java program generator"
- "python program generator"

В обзор не включены различные генераторы привязок к SQL таблицам (Spring Data JPA, jOOQ и подобные), шаблоны для языков разметки (jinja, Django Template Engine) в виду их узкой специализации. Были получены следующие результаты, соответствующие теме дипломной работы:

1.2 Automated C++ Program Generator using English Language Interface

В статье [2] описана программа, генерирующая код на C++ с помощью описания на английском языке. Из описания выделяются ключевые слова и параметры, которым сопоставляются один из множества поддерживаемых шаблонов и алгоритмов, в которые передаются параметры. Поддерживаются арифметические операции, числовые алгоритмы, строковые алгоритмы, алгоритмы над последовательностями и операции ввода-вывода.

1.3 Automatic code generation for C and C++ programming

В статье [3] описана программа, генерирующая код на C++ с помощью описания в виде блок-схем. Элементами блок-схемы является ввод-вывод, условные ветвления и циклы. Следствием этого является ограниченный набор поддерживаемых операций, но в то же время за счет низкоуровневого интерфейса данная программа может генерировать более сложные программы.

1.4 Csmith

Csmith — инструмент для генерации случайных программ на языке программирования C в соответствии со стандартом C99. Используется в тестировании компиляторов, благодаря нему получилось найти более 400 ошибок в компиляторах языка C которые не были известны до этого. Также поддерживает генерацию кода на C++. [4]

1.5 Liveness-Driven Random Program Generation (ldrgen)

Проект, основанный на идеях Csmith, также созданный для тестирования компиляторов. Основная идея - уменьшение количества “мертвого кода” при генерации, что позволяет добиться большего количества инструкций на строку кода и, соответственно, генерировать более компактные программы для тестирования. [5]

1.6 Yarpgen

Инструмент для генерации случайных программ на языке C для тестирования компиляторов. Для тестирования вычисляется хэш всех значений глобальных переменных программы после ее запуска. По сравнению с Csmith код, сгенерированный Yarpgen, более похож на написанный человеком, так как в некоторых случаях сначала генерируется более высокоуровневая модель, которая затем наполняется случайными данными. Также гарантируется отсутствие неопределенного поведения у сгенерированных программ. [6]

1.7 Deepsmith

Инструмент для генерации программ для библиотеки OpenCL на основе машинного обучения. Сгенерированный код похож на написанный человеком так как модель обучена на open-source коде с github. [7]

1.8 SL Random Program Generator

Инструмент для генерации случайных программ на языке Python. Можно настраивать количество инструкций и используемые в выражениях операторы. Имеется веб-версия, гле также можно найти исходны код и грамматику. [8]

1.9 Pyfuzz

Инструмент для генерации случайных программ на языке Python. Используется для тестирования инструментов компиляции и JIT-интерпретации Python-кода. [9]

Имеется веб-версия [10]

1.10 Сравнительный анализ найденных инструментов и статей

Сравнение аналогов будет проведено по следующим критериям:

- “Читаемость кода”, то есть похож ли сгенерированный код на написанный человеком

- Возможность расширения на разные языки программирования (расширяемость)
- Наличие интерфейса для взаимодействия
- Возможность настройки параметров генерации
- Поддержка рандомизации, в частности, возможность настроить начальное значение для генератора случайных чисел

Для статей ответы критерии будут проверяться из описания, так как код реализации отсутствует в открытом доступе.

Сравнение по данным критериям представлено в Таблице 1.

Инструмент	Читаемость	Расширяемость	Интерфейс	Настройка параметров	Рандомизация
Automated C++ Program Generator using English Language Interface	+	+	Natural language	+	-
Automatic code generation for C and C++ programming	+	+	Block-scheme	+	-
Csmith	-	-	CLI[1]	+	+
ldrgen	-	-	CLI	+	+
Yarpgen	-	-	CLI	+	+
Deepsmith	+	-	CLI	+	-
SL Random Program Generator	+	-	Web	+	+/-[2]
Pyfuzz	-	-	Web and CLI	+[3]	+[3]

Таблица 1: Сравнение аналогов.

[1] CLI = Command Line Interface (интерфейс командной строки)

[2] Отсутствует возможность задания seed для генератора случайных значений.

[3] Только в CLI

1.10.1 Результаты сравнения

Инструменты, описанные в статьях [2] и [3], имеют разный интерфейс, но оба имеют ограниченную параметризацию и генерируют читаемый код, однако не имеют поддержки рандомизации.

Csmith, ldrgen, и Yarpgen имеют схожий функционал и недостатки, однако среди них csmith имеет наиболее широкую степень параметризации, Yarpgen и ldrgen имеют меньшую возможность кастомизации.

Deepsmith благодаря машинному обучению генерирует код, максимально схожий с написанным человеком, однако, по этой же причине, обладает небольшой возможностью кастомизации и не поддерживает какую-либо рандомизацию.

SL Random Program Generator имеет удобный веб-интерфейс, но ограниченную возможность настройки и рандомизацию, так же очень ограничено количество поддерживаемых языковых конструкций языка Python.

Pyfuzz так же имеет веб интерфейс, однако в нем совсем отсутствует возможность настройки и рандомизации. В CLI такая возможность присутствует, однако получившиеся программы все же используются для тестирования компиляторов и интерпретаторов, поэтому код получется плохо читаемым для человека.

Глава 2. Разработка инструмента генерации программ

Сравнительный анализ существующих решений для генерации программ показал, что инструменты, используемые на практике, не подходят для учебных целей. Поэтому было принято решение разработать собственную систему генерации программ для учебных задач.

2.1 Требования к системе генерации

Разрабатываемый инструмент должен обладать следующими возможностями:

- Возможность генерации базовых элементов языка программирования;
- Поддержка генерации кода на разных языках, чтобы иметь возможность использовать данный инструмент в разных обучающих курсах;
- Расширяемость, что включает в себя:
  - Гибкую систему создания шаблонов для генерации задач;
  - Возможность настройки параметров генерации;
  - Высокую вариативность задач, поддержку рандомизации отдельных элементов кода

2.2 Схема генерации программ

На схеме 1 предоставлена схема генерации кода программы.

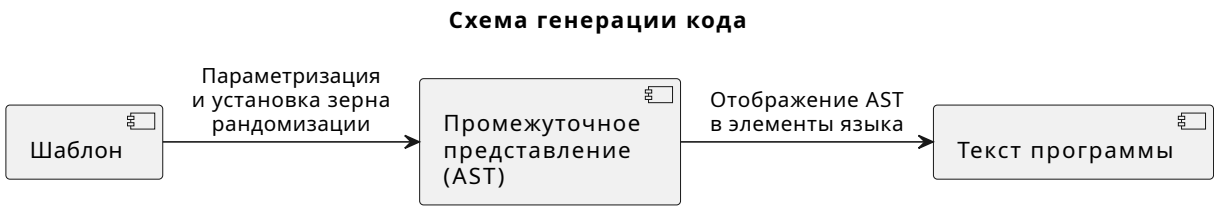

Рис. 1: Схема генерации кода программы

Разберём ее подробнее. На первом шаге выбирается шаблон, из которого будет генерироваться программа. Подробнее о шаблонах можно прочитать в разделе 2.3, на текущем этапе нам достаточно знать, что шаблон — это некая структура, в которую можно подставить начальное значение для генератора псевдослучайных чисел (`seed`) и словарь атрибутов, и получить промежуточное представление, о котором речь пойдет ниже. `seed` представляет собой целое 64-битное число, словарь атрибутов сопоставляет строковому ключу список строковых значений, некоторые из которых могут быть преобразованы в числа

Промежуточное представление является по сути расширением абстрактного синтаксического дерева (AST)

«Дерево абстрактного синтаксиса (ДАС) — в информатике конечное помеченное ориентированное дерево, в котором внутренние вершины сопоставлены (помечены) с операторами языка программирования, а листья — с соответствующими операндами. Таким образом, листья являются пустыми операторами и представляют только переменные и константы.» [11]

В отличие от чистого AST промежуточное представление, используемое в проекте, хранит некоторую информацию о синтаксисе языка (или группы языков), в которые оно в дальнейшем будет интерпретировано. К примеру, в данном дереве могут содержаться скобки в арифметических выражениях, также промежуточное представление хранит метку (тэг), соответствующую группе языков.

На последнем этапе происходит преобразование промежуточного представления в код. Во время преобразования элементам промежуточного предоставление сопоставляются элементы синтаксиса конкретного языка программирования.

2.3 Шаблоны программ

2.3.1 Общие единицы кода для разных языков

При генерации кода на разных языках можно выделить конструкции, которые имеют схожую семантику в разных языках, но, при этом, могут отличаться синтаксически. Такими конструкциями являются к примеру:

- Арифметические выражения
- Условные ветвления (`if...else`)
- Множественный выбор/сопоставление с образцом (`switch...case, match, when, case...of`)
- Циклы (`for, while`)
- Объявление и инициализация переменных
- Объявление, определение и вызов функций
- Блоки кода
- Инструкции подключения модулей/библиотек (`import, include`)
- Строковые и числовые литералы
- Объявление классов и структур
- ...

Для каждой из этих конструкций будет достаточно одной сущности в шаблоне, которую смогут образовать преобразователи промежуточного предоставления для каждого конкретного языка.

2.3.2 DSL

Для создания шаблонов программ в дипломном проекте используется язык программирования Kotlin [12], а конкретно одна из особенностей языка под названием Kotlin DSL [13].

«Предметно-ориентированный язык (англ. domain-specific language, DSL — «язык, специфический для предметной области») — компьютерный язык, специализированный для конкретной области применения (в противоположность языку общего назначения, применимому к широкому спектру областей и не учитывающему особенности конкретных сфер знаний)». [14]

Используемый в проекте DSL является внутренним, то есть написан на языке общего назначения (в данном случае Kotlin) и имеет точно такой же синтаксис. *TODO(пример DSL)*

Благодаря возможностям Kotlin DSL, можно воспроизводить в шаблоне древовидную структуру, где параметры функции — это параметры в вершине AST, а последний аргумент (ламбда-функция, переданная за пределами скобок) описывает поддерево.

2.3.3 Шаблон AST

Во время выполнения DSL преобразуется в «шаблон AST» — древовидную структуру данных, которая похожа на AST. В ней могут присутствовать специальные вершины, обозначающие случайное значение или значение какого-либо атрибута. При подстановке в нее `seed` и `attributes` она преобразуется в промежуточное представление.

(TODO: добавить шаблонное дерево задачи из предыдущего пункта)

2.4 Примеры задач

(TODO: добавить примеры задач)

2.5 Архитектура системы

Для поддержки одновременной работы с несколькими пользователями и возможности добавления шаблонов и валидации ответов студентов разработанная система имеет клиент-серверную архитектуру с отдельными микросервисами для некоторых компонент. Благодаря этому удалось добиться конфиденциальности правильных ответов и контроля над управлением шаблонов.

Клиентская часть представляет собой веб-сайт с помощью которого пользователь может делать запросы на получение текста или картинки кода по шаблону (задаче). Также имеется возможность делать запросы напрямую к API в формате JSON [15].

Серверная часть состоит из нескольких компонент: непосредственно веб-сервер, система генерации кода и система проверки ответов студентов.

В отдельный микросервис выделена система хранения текстов и изображений программ, состоящая из базы данных, находящейся в изолированном окружении.

Схема архитектуры показана на изображении 2, ниже подробнее описаны отдельные компоненты.

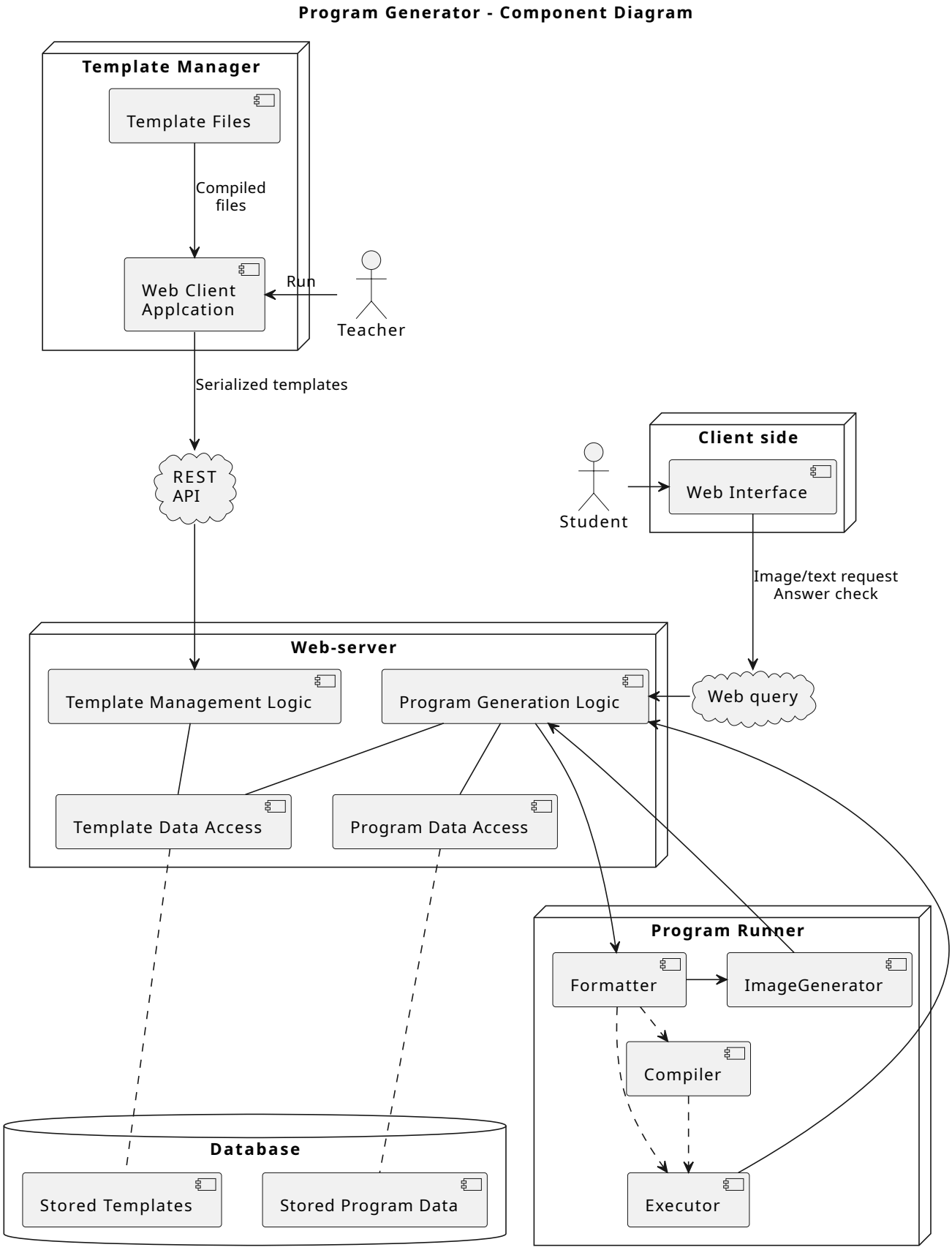

Рис. 2: Архитектура системы генерации программ

2.6 Компоненты системы

2.6.1 Веб-сервер

Веб-сервер (Web-server на схеме) разделен на две логические части — управление шаблонами (Template Management Logic), доступная только преподавателям или администраторам, и генерация программ (Program Generation Logic). Первая отвечает за добавление и удаление шаблонов программ, вторая — за создание и получение изображения и текста программы, а так же за верификацию ответов на задачу.

2.6.2 Исполнитель программ

Исполнитель программ (Program runner на схеме) отвечает за обработку сгенерированного кода. Он содержит в себе инструмент форматирования кода, инструмент генерации изображения кода, а также окружение (компилятор и/или интерпретатор), необходимое для выполнения программы. С помощью данного компонента осуществляется генерация изображения и получение вывода программы при ее запуске.

Исполнитель вынесен в отдельный компонент так как ему необходимо специальное окружение (установленные компиляторы, интерпретаторы, средства форматирования).

2.6.3 База данных

Для поддержки работы с несколькими пользователями необходимо хранить шаблоны и данные о сгенерированных программах в базе данных.

В базе сгенерированных программ (Stored Program Data) хранятся параметры генерации, которые, вместе с идентификатором задачи, выступают ключом. Также в ней хранятся текст программы, изображение и вывод программы при запуске.

В базе шаблонов (Stored Templates) хранятся шаблоны программ, тэг, соответствущий языку или группе языков, для которых написан этот шаблон, и имя шаблона (задачи), которое является ключом.

2.6.4 Сторона клиента

На клиентской стороне пользователь может как взаимодействовать напрямую с сервером, делая запросы через строку браузера и получая ответ в формате HTML, так и через специальные образовательные платформы по типу Moodle [16].

2.6.5 Менеджер шаблонов

Для создания и управления шаблонами программ на стороне клиента используется отельная программа — менеджер шаблонов (Template Manager на схеме). С помощью нее при добавлении текст шаблона преобразуется в машиночитаемый вид *(TODO: вставить ссылку на детали реализации)* и затем, вместе с прочей необходимой информацией (названием, тэгом) отправляется на сервер для сохранения в базу шаблонов (см. 2.6.3).

Глава 3. Реализация инструмента генерации программ

3.1 Используемые технологии

Для разработки инструмента был выбран язык программирования Kotlin [12]. Это язык программирования, разработанный компанией JetBrains в 2010 году. Основными преимуществами этого языка программирования являются:

- Кроссплатформенность
- Возможность интеграции с различными популярными системами сборки (Maven, Gradle, etc.)
- Возможность интеграции с java без переписывания имеющегося кода

Для автоматизации развертывания системы генерации, поддерживания окружения для хранилища и системы проверки используются технологии Docker [17] и docker-compose [18]. Благодаря Docker можно создать изолированное окружение (контейнер) для компонента, а docker-compose позволяет объединять контейнеры в единую локальную сеть. (*TODO: описать для каких компонентов сделаны контейнеры*)

Для компонента сервера было принято решение использовать библиотеку Ktor [19], так как она позволяет быстро создать веб-сервер с нужной функциональностью и имеет простой и элегантный API.

(*TODO: описание методов API*)

(*TODO: примеры кода*)

Генерация изображений реализована с помощью библиотеки `javax.imageio` [20]. По тексту генерируется изображение в формате png.

Глава 4. Заключительный раздел с основными результатами

4.1 Подраздел

4.2 Подраздел

Заключение

Заключение должно подводить итоги работы и содержать информацию о полученных в рамках работы результатах.

Список литературы

[1] А Хафизова и М Заславский. «Генератор случайных программ как инструмент обучения программированию». В: *СБОРНИК ДОКЛАДОВ СТУДЕНТОВ И АСПИРАНТОВ НА КОНФЕРЕНЦИИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА*. 2019, с. 191.

[2] Ambuj Kumar и Saroj Kaushik. «Automated C++ Program Generator using English Language Interface». В: ().

[3] S. Patade и др. «AUTOMATIC CODE GENERATION FOR C AND C++ PROGRAMMING». В: *IRJET* 08 (05 2021), с. 4732—4736.

[4] Xuejun Yang и др. *Csmith*. URL: `https://embed.cs.utah.edu/csmith/` (дата обр. 10.05.2022).

[5] Gergö Barany. «Liveness-driven random program generation». В: *International Symposium on Logic-Based Program Synthesis and Transformation*. Springer. 2017, с. 112—127.

[6] Intel. *Yarpgen*. URL: `https://github.com/intel/yarpgen` (дата обр. 10.05.2022).

[7] Chris Cummins и др. «Compiler fuzzing through deep learning». В: *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*. 2018, с. 95—105.

[8] Ariel Baruch. *SL Random Program Generator*. URL: `https://www.cs.bgu.ac.il/~arielbar/sl/#/code` (дата обр. 10.05.2022).

[9] Steven Myint. *pyfuzz*. URL: `https://github.com/myint/pyfuzz` (дата обр. 10.05.2022).

[10] Steven Myint. *Random Python Program Generator*. URL: `https://www.4geeks.de/cgi-bin/webgen.py` (дата обр. 10.05.2022).

[11] *AST*. URL: `https://en.wikipedia.org/wiki/Abstract_syntax_tree` (дата обр. 15.05.2022).

[12] Jemerov D. и Isakova S. *Kotlin in action*. Manning Publications Company, 2017.

[13] Jemerov D. и Isakova S. «Kotlin in action». В: Manning Publications Company, 2017. Гл. 11, с. 346—380.

[14] *Предметно-ориентированный язык*. URL: https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D0%B4%D0%BC%D0%B5%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA (дата обр. 15.05.2022).

[15] ECMA International. *Standard ECMA-404. The JSON Data Interchange Format*. 2017.

[16] *Moodle*. URL: https://moodle.org/?lang=ru (дата обр. 15.05.2022).

[17] *Docker*. URL: https://www.docker.com/ (дата обр. 10.05.2022).

[18] *docker-compose*. URL: https://docs.docker.com/compose/ (дата обр. 10.05.2022).

[19] JetBrains. *Ktor*. URL: https://ktor.io/ (дата обр. 10.05.2022).

[20] Oracle. *javax.imageio*. URL: https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/javax/imageio/ImageIO.html (дата обр. 10.05.2022).