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:
профессор, д.ф.-м.н. А. А. Выбегалло

Рецензент:
старший разработчик ООО «Рога и копыта»
А. И. Привалов

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 моей выпускной работы является создать расширяемый генератор учебных задач для курсов 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 код для создания учебных задач.

1. Обзор и сравнение существующих генераторов программного кода

1.1. Понятие генерации программного кода

<Написать определение и что-то еще про генерацию>

Был произведен поиск в поисковых системах “Google” и “Google Scholar” по ключевым словам “C++ program generator”, “program generator”, “random 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 моего диплома:

1.2. Automated C++ Program Generator using English Language Interface

В статье [1] 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 [2] 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++. [3]

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. [4]

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. [5]

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. [6]

1.8. 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

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

Инструмент	Чита-емость	Расширя-емость	Интер-фейс	Настройка парамет-ров	Рандо-мизация
Automated C++ Program Generator using English Language Interface	+	+	Natural language	+	-
Automatic code generation for C and C++ programming	+	+	Block-scheme	+	-
Csmith	-	-	CLI	+	+
ldrgen	-	-	CLI	+	+
Yarpgen	-	-	CLI	+	+
Deepsmith	+	-	CLI	+	-

Таблица 1: Сравнение аналогов.

Инструменты, описанные в статьях [1] и [2], 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.

2. Основной раздел

2.1. Подраздел

2.2. Подраздел

2.3. Подраздел

3. Основной раздел

3.1. Подраздел

3.2. Подраздел

3.3. Подраздел

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] Ambuj Kumar, Saroj Kaushik. Automated C++ Program Generator using English Language Interface. ICCS-2005, December 2005.

[2] S. Patade, P. Patil, A. Kamble , M. Patil. AUTOMATIC CODE GENERATION FOR C AND C++ PROGRAMMING. IRJET, May 2021.

[3] Csmith. URL: `https://embed.cs.utah.edu/csmith/` (дата обр. 19.04.2022).

[4] Gergö Barany. Liveness-Driven Random Program Generation, 2017. hal-01658563.

[5] Yarpgen. URL: `https://github.com/intel/yarpgen` (дата обр. 19.04.2022).

[6] Cummins, Chris and Petoumenos, Pavlos and Murray, Alastair and Leather, Hugh. Compiler fuzzing through deep learning. Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis, 2018. (pp. 95-105)