DOI 10.37882/2223-2966.2024.04.30

РАЗРАБОТКА ПРОГРАММЫ ГЕНЕРАЦИИ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ ДЛЯ ЗАДАЧ ОБУЧЕНИЯ В НАПИСАНИИ ЗАГОЛОВКОВ ФУНКЦИЙ

DEVELOPMENT OF A REGULAR EXPRESSION GENERATION PROGRAM FOR LEARNING TASKS IN WRITING FUNCTION HEADERS

V. Sukhoverkhov
I. Gulyaev
I. Shabanova

Summary. This article discusses the methodology of developing a program for the automatic generation of tasks aimed at teaching students the skills of writing function headers using regular expressions. The function header is a key element of the program code that defines its interface and functionality. Regular expressions are a powerful tool for working with text, allowing you to find and analyze patterns. The program being developed in the article will create a variety of tasks for compiling function headers based on specified templates using regular expressions. This approach will allow students to practice understanding the syntax and structure of function headers, which will significantly increase their level of programming skills.

Keywords: program development, generation, regular expressions, function headers, training.

Суховерхов Владислав Вячеславович
Волгоградский государственный технический университет
suhoverhov.vladislav1804@gmail.com
Гуляев Иван Владимирович
Волгоградский государственный технический университет
aioki@outlook.com
Шабанова Ирина Николаевна
Волгоградский государственный технический университет
shabanovaira34@gmail.com

Аннотация. В данной статье рассматривается методика разработки программы для автоматической генерации задач, направленных на обучение студентов навыкам написания заголовков функций с использованием регулярных выражений. Заголовок функции — ключевой элемент программного кода, определяющий её интерфейс и функциональность. Регулярные выражения представляют собой мощный инструмент для работы с текстом, позволяющий находить и анализировать шаблоны. Программа, разрабатываемая в статье, будет создавать разнообразные задачи по составлению заголовков функций на основе заданных шаблонов с использованием регулярных выражений. Этот подход позволит студентам практиковаться в понимании синтаксиса и структуры заголовков функций, что существенно повысит их уровень владения навыками программирования.

Ключевые слова: разработка программы, генерация, регулярные выражения, заголовки функций, обучение.

Вступление

Регулярные выражения играют важную роль в мире программирования, обеспечивая эффективные инструменты для обработки и анализа текстовых данных. Однако составление регулярных выражений может быть сложным и трудоемким процессом, особенно для новичков. В контексте образовательного процесса, где студенты изучают основы программирования, составление регулярных выражений вручную может представлять значительные трудности.

Ручное составление регулярных выражений сталкивается с рядом проблем. Во-первых, это сложный процесс, требующий глубокого понимания синтаксиса и правил составления выражений. Во-вторых, такой подход подразумевает ограниченную гибкость и недостаточную покрытость возможных вариантов. Кроме того, обновление и поддержка регулярных выражений, особенно при изменении требований к задачам, также может быть трудоемким и ошибочным процессом [2, 4, 9].

В данной статье мы представляем программу для генерации регулярных выражений, разработанную специально для использования преподавателями в образовательных целях. Эта программа предоставляет простое и эффективное решение для проверки ответов студентов по написанию прототипов функций. Вместо ручного составления выражений, преподаватели могут использовать эту программу для генерации разнообразных шаблонов, учитывающих различные типы параметров функций и варианты их расположения. Такой подход обеспечивает более точную и полную проверку ответов студентов, сокращает время, затрачиваемое на подготовку и обновление задач и упрощает процесс обучения.

Проблема ручного составления регулярных выражений

Регулярные выражения представляют собой мощный инструмент для работы с текстом, который позволяет выполнять поиск, анализ и манипуляции строками в текстовых данных. Они используются для определения ша-

блонов и правил, по которым производится поиск соответствий в тексте. Это позволяет эффективно извлекать информацию из текстовых файлов, в том числе и для автоматизации обработки данных. Однако, составление их вручную может стать затруднительным и трудоемким процессом, особенно для начинающих программистов. Вот некоторые проблемы, с которыми сталкиваются люди при ручном создании регулярных выражений [1, 3]:

1. сложность синтаксиса — регулярные выражения имеют свой собственный синтаксис, который может быть непривычным для новичков. Этот синтаксис включает в себя специальные символы и конструкции, которые нужно понимать и использовать правильно;
2. трудности в отладке — после написания регулярного выражения может потребоваться тщательная отладка, чтобы убедиться, что оно работает правильно. Ошибки в регулярных выражениях могут быть трудно обнаружить и исправить, особенно для тех, кто только начинает изучать их;
3. неоптимальность — некоторые регулярные выражения могут быть неоптимальными с точки зрения производительности, что может привести к медленной обработке текстовых данных или неправильным результатам.
4. ограниченная гибкость — ручное создание регулярных выражений ограничено возможностями программиста и его пониманием синтаксиса. Это может привести к узкому кругу вариантов проверки и обработки текста;
5. трудности в поддержке и обновлении — при изменении требований или появлении новых данных может потребоваться пересмотр регулярных выражений. Этот процесс может быть сложным и трудоемким, особенно при наличии большого количества регулярных выражений в проекте.

Методика генерации задач обучения

Для создания регулярного выражения, которое будет соответствовать прототипу функции, необходимо учитывать синтаксические правила языка программирования, а также возможные вариации форматов, которые могут встречаться в заголовках функций [5, 7].

Программа должна позволять составить прототип функции оперируя концептуальными типами данных без привязки к конкретному языку программирования (рис. 1).

Каждый прототип функции можно свести к единому формату, что в конечном итоге сводится к регулярному выражению, которое соответствует прототипу функции, шаблон которого несложно описать (рис. 2).

^\s* — начало строки с возможными пробелами в начале
\b (тип возвращаемого значения) \b — тип возвращаемого значения, заключённый в границы слова
\s+ — один или более пробелов
\b (имя функции) \b — имя функции, заключённое в границы слова
\s*\(— возможные пробелы, за которыми следует открывающая скобка
(?: …)? — необязательная группа, содержащая

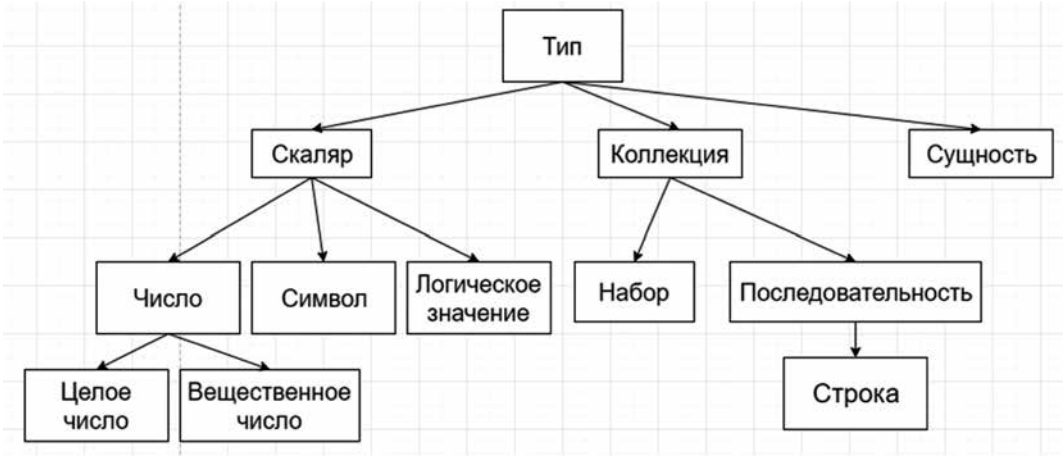

Рис. 1. Концептуальные типы данных

```
тип_возвращаемого_значения имя_функции(тип_параметра1 аргумент1, тип_параметра2 аргумент2, ...)

⇓

^\s*\b(тип_возвращаемого_значения)\b\s+\b(имя_функции)\b\s*\((?:\s*\b(тип_параметра) ↩
\b\s+\b(аргумент)\b\s*(?:,\s*\b(тип_параметра)\b\s+\b(аргумент)\b\s*)*)?\)\s*;?$
```

Рис. 2. Универсальное описание регулярного выражения

\s*\b (тип параметра)\b\s+\b(аргумент)\b — тип параметра и его имя, разделённые одним или более пробелами
(?:,\s*\b(тип параметра)\b\s+\b(аргумент)\b\s*)* — необязательная последовательность параметров, разделённых запятыми
?\)— возможная закрывающая скобка
\s*;?$— возможные пробелы, за которыми следует точка с запятой и/или конец строки

При составлении прототипа функции для генерации задач обучения необходимо учитывать, что порядок параметров в функции может быть произвольным. Это означает, что при проверке ответов студентов необходимо учитывать возможность передачи аргументов в любом порядке.

Кроме того, способы передачи аргументов в функцию могут различаться. Например, студенты могут передавать два или более параметров единым массивом, либо использовать единую структуру для передачи всех параметров. Это требует учета различных вариантов ответов при генерации регулярных выражений для проверки правильности написания заголовков функций.

Такой подход позволяет обеспечить более гибкую и адаптивную систему проверки, которая учитывает разнообразие способов передачи аргументов и не зависит от конкретного порядка параметров в функции.

Разработка программы для генерации задач

1. Генерация регулярных выражений

Программа должна содержать модуль для генерации регулярных выражений на основе заданных параметров функции [6,8]. Этот модуль может использовать различные алгоритмы и методы для создания разнообразных комбинаций параметров:

1. полный перебор — метод полного перебора позволяет создать все возможные комбинации параметров функции. Этот метод гарантирует, что будут рассмотрены все варианты, но может быть неэффективным при большом количестве параметров;
2. генерация случайных комбинаций — метод генерации случайных комбинаций параметров позволяет создать разнообразные наборы аргументов для функции. Это позволяет получить разнообразные тестовые наборы и проверить функцию на различных входных данных;
3. эвристические алгоритмы — использование эвристических алгоритмов для создания комбинаций параметров, основанных на определенных правилах или эвристиках. Например, можно определить некоторые паттерны или шаблоны для комбинирования параметров и использовать их при генерации;
4. генетические алгоритмы — генетические алгоритмы могут использоваться для эффективного поиска оптимальных или приближенных комбинаций параметров. Это позволяет создать комбинации, которые могут обеспечить лучшее покрытие различных случаев использования функции;
5. методы машинного обучения — использование методов машинного обучения для анализа и генерации комбинаций параметров на основе предоставленных данных о функции. Это может включать в себя обучение моделей на основе существующих примеров или использование методов кластеризации для определения различных групп параметров.

2. Графический интерфейс пользователя (GUI)

Для удобства использования программа может иметь GUI, позволяющий вводить параметры функции и получать соответствующие регулярные выражения.

3. Алгоритмический модуль для обработки регулярных выражений

Программа должна включать модуль для обработки и анализа сгенерированных регулярных выражений, чтобы убедиться, что они соответствуют требованиям задачи.

4. Поддержка различных типов параметров

Программа должна быть способной обрабатывать различные типы параметров функций, включая простые типы данных (например, целые числа, строки) и более сложные структуры данных (например, коллекции, структуры).

5. Модуль для отображения результатов

Программа должна предоставлять пользователю удобный способ просмотра сгенерированных регулярных выражений.

Эффективность программы в обучении студентов

Эффективность программы в обучении студентов зависит от нескольких факторов:

1. Понятность интерфейса:
- графический интерфейс должен быть интуитивно понятным и удобным в использовании для студентов. Чем проще и понятнее интерфейс, тем легче студентам будет освоить программу.
2. Генерация разнообразных задач:
- программа должна быть способна генерировать разнообразные задачи по написанию заголовков функций с использованием регулярных выраже-

ний. Это поможет студентам получить опыт в работе с различными типами данных и структурами.
3. Подробные объяснения:
- помимо генерации задач, программа может предоставлять подробные объяснения и комментарии к каждой задаче, помогая студентам понять, как правильно составить заголовок функции с учетом конкретного входного набора данных.
4. Возможность самостоятельной проверки:
- хорошая программа должна предоставлять студентам возможность самостоятельно проверять свои ответы на задачи. Это позволяет им моментально узнавать свои ошибки и исправлять их.
5. Мотивационные элементы:
- возможно, добавление игровых элементов или системы наград может сделать обучение более увлекательным и мотивирующим для студентов.
6. Отслеживание прогресса:
- программа может вести отслеживание прогресса студентов, показывая им, какие задачи они уже выполнены, и какие имеются задачи в запасе. Это поможет студентам оценить свой прогресс и увидеть свои успехи.
7. Обратная связь:
- важно предоставлять студентам обратную связь о том, где они допускают ошибки и как они могут их исправить. Это поможет им понять свои слабые места и сосредоточиться на их улучшении.

Эффективность программы в обучении студентов

Программа может быть эффективным инструментом для обучения студентов, обучающихся работе с регулярными выражениями и написанию заголовков функций. Вот несколько способов, как программа может быть полезной:

1. Практическое применение:
- студенты могут использовать программу для практического изучения применения регулярных выражений в реальных задачах программирования.
2. Понимание структуры функций:
- генерация заголовков функций с использованием регулярных выражений поможет студентам лучше понять структуру функций и правильное именование параметров.
3. Эксперименты с различными комбинациями:
- студенты могут экспериментировать с различными комбинациями параметров функций и анализировать, как это влияет на структуру регулярных выражений.

Перспективы развития и дальнейшие исследования

1. Улучшение интерфейса:
- развитие более интуитивного и информативного пользовательского интерфейса может улучшить опыт использования программы
2. Расширение функциональности:
- добавление поддержки более широкого спектра типов данных и функций может сделать программу более гибкой и полезной для студентов.
3. Исследование методов обучения:
- исследование эффективности программы в обучении и ее влияния на усвоение материала студентами может помочь определить оптимальные методы обучения.
4. Интеграция с образовательными платформами:
- интеграция программы с существующими образовательными платформами может облегчить доступ студентов к обучающему материалу.
5. Расширение функциональности для продвинутых пользователей:
- для более опытных пользователей можно добавить дополнительные функции и параметры, позволяющие им более глубоко изучать регулярные выражения и функциональное программирование.

Заключение

В заключение, разработанная программа представляет собой мощный инструмент для обучения студентов работе с регулярными выражениями и пониманию структуры заголовков функций [10]. Ее гибкость и функциональность делают ее идеальным инструментом как для начинающих программистов, так и для более опытных пользователей, желающих углубить свои знания в области регулярных выражений и функционального программирования.

В ходе разработки была реализована структура программы, включающая генерацию регулярных выражений, создание графического интерфейса пользователя (GUI) с использованием библиотеки PyQt, а также модуль для обработки регулярных выражений и отображения результатов. Использование архитектурного паттерна Стратегия позволило реализовать различные способы вычисления комбинаций параметров функций, что значительно повысило гибкость и расширило функциональность программы.

Эффективность программы в обучении студентов подтверждается ее способностью предоставлять студентам практический опыт работы с регулярными выражениями, а также понимание структуры и назначения заголовков функций. Перспективы развития программы включают в себя улучшение интерфейса, расширение функциональности, дальнейшие исследования методов обучения и интеграцию с образовательными платформами.

В целом, программа представляет собой важный инструмент для обучения и самостоятельного изучения темы регулярных выражений и функционального программирования, и ее развитие и совершенствование могут значительно способствовать улучшению качества образования в области программирования.

ЛИТЕРАТУРА

1. Батура Т.В., Бакиева А.М. Методы и системы автоматического реферирования текстов //Новосибирск: ИПЦ НГУ. — 2019 — 59 с.
2. Волкова Г.А. Анализ методов выявления заимствований программного кода //Новые информационные технологии в автоматизированных системах. — 2011. — №. 14. — С. 10–21.
3. Гниденко И., Федоров Д., Павлов Ф. Технология разработки программного обеспечения // Учебное пособие для СПО. — Litres — 2017 — 219 с.
4. Душейко А.Ю. и др. Генерация лидов новостных текстов с помощью нейронной сети ruGPT-3: магистерская диссертация по направлению подготовки: 45.04. 03-Фундаментальная и прикладная лингвистика. — 2022.
5. Канжелев С.Ю., Шалыто А.А. Автоматическая генерация автоматного кода //Информационно-управляющие системы. — 2006. — №. 6. — С. 35–42.
6. Карпов М.Г. и др. Извлечение структурированной информации из текстов исковых заявлений //Математическое и информационное моделирование. — 2020. — С. 183–191.
7. Леонов Ф.В., Челпанов А.Д., Юганссон А.Н. Программирование на языке Python для решения задач информационной безопасности. — 2021.
8. Пашков П.М., Печень О.А. Применение регулярных выражений для создания каталога интернет-магазина // Вестник НГУЭУ. — 2014. — №. 3. — С. 329–336.
9. Сизых Я. В. Электронная система адаптивного обучения: дис. — Сибирский федеральный университет, 2016.
10. Туракулова А.И. Возможности нового объектно-ориентированного языка программирования ObjectScript //Современные информационные технологии и ИТ-образование. — 2015. — Т. 2. — №. 11. — С. 104–108.

© Суховерхов Владислав Вячеславович (suhoverhov.vladislav1804@gmail.com); Гуляев Иван Владимирович (aioki@outlook.com);
Шабанова Ирина Николаевна (shabanovaira34@gmail.com)
Журнал «Современная наука: актуальные проблемы теории и практики»