Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Волгоградский государственный технический университет

Факультет Электроники и вычислительной техники
Кафедра Программное обеспечение автоматизированных систем

Согласовано

_____
(должность гл. специалиста предприятия)

_____ _____
(подпись) (инициалы, фамилия)
«_____» _____ 2017

Утверждаю
Зав. кафедрой ПОАС, д.т.н., проф.

_____ А. М. Дворянкин
(подпись) (инициалы, фамилия)
«_____» _____ 2017

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к выпускной работе бакалавра на тему
(наименование вида работы)
Поддержка перечислений для языка C++ в типе вопроса CorrectWriting

Автор _____ Клевцов Вадим Александрович
(подпись и дата подписания) (фамилия, имя, отчество)

Обозначение ВРБ-40-461-806-10.19-09.03.04-02-15-81
(код документа)

Группа ПрИн-466
(шифр группы)

Направление 09.03.04 Программная инженерия
(код по ОКСО, наименование направления, программы)

Руководитель работы _____ к.т.н О. А. Сычев
(подпись и дата подписания) (инициалы и фамилия)

Консультанты по разделам:

(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)
(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)
(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)

Нормоконтролер _____ О. Н. Ляпина
(подпись и дата подписания) (инициалы и фамилия)

Волгоград, 2017

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Волгоградский государственный технический университет

Кафедра Программное обеспечение автоматизированных систем

Утверждаю
Зав. кафедрой ПОАС, д.т.н., проф.

_____ А. М. Дворянкин
(подпись) (инициалы, фамилия)
«_____» _____ 2017

Задание на выпускную работу бакалавра
(наименование вида работы)
Студент Клевцов Вадим Александрович
(фамилия, имя, отчество)
Код кафедры 10.19 Группа ПрИн-466

Тема Поддержка перечислений для языка C++ в типе вопроса CorrectWriting
Утверждена приказом по университету от «17» октября 2014 № 1529–ст
Срок представления готовой работы _____
(дата, подпись студента)
Исходные данные для выполнения работы
задание, выданное научным руководителем с кафедры ПОАС, утвержденное приказом ректора

Содержание основной части пояснительной записки

Введение

1 Анализ состояния поддержки перечислений в типах вопросов системы дистанционного образования Moodle

Цель и задачи исследования

2 Разработка модуля поддержки перечислений

Выводы

3 Оценка разработанного решения

Выводы

Заключение

Список использованных источников

Приложение А - Техническое задание

Перечень графического материала

1) 1: Название работы
2) 2-3: Актуальность
3) 4: Постановка задачи
4) 5: Перечисление
5) 6: Цель и задачи
6) 7: Исследование стандартов языков C/C++
7) 8: Требования к графическому интерфейсу
8) 9: Демонстрация интерфейса
9) 10: Алгоритм определения порядка элементов перечисления, дающего наибольшую LCS
10) 11-13: Пример
11) 14: Диаграмма классов
12) 15: Тестирование
13) 16-17: Интеграция алгоритма в плагин CorrectWritting
14) 18: Оценка уменьшения трудоемкости
15) 19: Выводы
16) 20: Апробация
17) 21: Благодарность
18) 22: Построенные LCS

Руководитель работы _____ к.т.н О. А. Сычев
(подпись и дата подписания) (инициалы и фамилия)

Консультанты по разделам:

(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)
(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)
(краткое наименование раздела)	(подпись и дата подписания)	(инициалы и фамилия)

Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
Волгоградский государственный технический университет
Кафедра «Программное обеспечение автоматизированных систем»

Утверждаю
Зав. кафедрой ПОАС, д.т.н., проф.

_____ А. М. Дворянкин
(подпись) (инициалы, фамилия)
«_____» _____ 2017

Поддержка перечислений для языка C++ в типе вопроса CorrectWriting

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

ВРБ-40-461-806-10.19-09.03.04-02-15-81

Листов 37

Научный руководитель
к.т.н, доц. каф. ПОАС
_____ О. А. Сычев
«_____» _____ 2017

Нормоконтролер
ст. преп. каф. ПОАС
_____ О. Н. Ляпина
«_____» _____ 2017

Исполнитель
студент группы ПрИн-466
_____ В. А. Клевцов
«_____» _____ 2017

Волгоград, 2017

ВРБ-40-461-806-10.19-09.03.04-02-15-81

Аннотация

Документ представляет собой пояснительную записку к выпускной работе бакалавра на тему «Поддержка перечислений для языка C++ в типе вопроса CorrectWriting», выполненную студентом группы ПрИн-466, Клевцовым Вадимом Александровичем.

В данной работе было рассмотрено применение определения перечислений к вычислению редакционного расстояния и обнаружению синтаксических ошибок в ответе в свободной форме студента при применении его в изучении языков программирования.

Объём пояснительной записки составил 37 страниц и включает 7 рисунков и 3 таблицы.

Ключевые слова: система дистанционного образования, Moodle, мудл, CorrectWriting, перечисления, enumerations.

Содержание

Введение

На данный момент в учебном процессе широко используются системы дистанционного образования (далее СДО), такие как Moodle, Blackboard и другие. Эти системы используются для обеспечения студентов учебно-методическими материалами, информацией о событиях, происходящих в рамках учебного процесса, а также организации удалённой связи преподавателя со студентом.

Одной из самых важных функций систем дистанционного образования является возможность проведения в них автоматизированного тестирования. Автоматизированное тестирование, в качестве инструмента контроля знаний студента в рамках изучения некоторых дисциплин позволяет существенно снизить затраты аудиторного времени на проверку знаний студента, и повысить качество данной проверки за счет применения различных методик.

СДО Moodle является одной из широко используемых систем дистанционного образования. Данная система широко применяется во множестве высших учебных заведениях в мире.

В данной системе присутствует возможность построения в ней учебных курсов в рамках интерфейса, понятного преподавателю и студенту, а также возможность организации в ней автоматизированного тестирования в рамках изучения определённой дисциплины, что является её большим достоинством.

В данный момент внедрение автоматизированного тестирования при изучении дисциплины «Основы программирования», которое проводилось с использованием данной системы, позволило уменьшить затраты аудиторного времени на проверку базовых знаний студента по теме, перед более точной проверкой, в рамках решения более сложных инженерных задач, а также повысить объективность оценивания и увеличить подготовленность студента к занятиям.

Демонстрационные и тренировочные тесты представляют собой тесты, состоящие из нескольких заданий, которые проходятся студентом во время, отведённое для самоподготовки. Как правило, данные виды

тестов позволяют студенту сразу после ввода ответа на вопрос проверить, насколько его ответ близок к ответу учителя и на основе этого провести самопроверку, после чего он может исправить ошибку. При этом время выполнения данных тестов чаще всего не ограничивается. Целью выполнения тренировочного теста, для студента, является выявление пробелов в собственных знаниях. Для этого студенту часто необходимо иметь информацию о том, ошибка из какой части изучаемого материала была совершена.

Прохождение демонстрационного и тренировочного тестов позволяет студенту лучше подготовиться к сдаче контрольного теста, в процессе ответа которого присутствует ограничение по времени в объеме 30 минут. Также результат теста нельзя изменять после сдачи.

Значительную долю заданий, в рамках данной дисциплины, составляют задания, связанные с тем, что студенту предлагается написать верную синтаксическую конструкцию в рамках изучаемого языка.

Одной из основных проблем при использовании автоматизированного тестирования в рамках данной дисциплины является то, что при наличии нескольких верных ответов, вследствие наличия лексем, порядок которых в ответе не важен, преподавателю приходится вводить все варианты ответа, а также вводить описание каждой лексемы в каждом ответе.

Данная работа посвящена созданию системы, позволяющей учитывать последовательности лексем порядок которых не важен.

1 Анализ состояния поддержки перечислений в типах вопросов системы дистанционного образования Moodle

1.1 Обобщенное описание проблемы перечислений

Основным достоинством формальных языков - жестко заданный порядок лексем в выражении. С другой стороны из жесткого порядка бывают исключения. Такие исключения чаще являются перечислениями, то есть последовательностью элементов отделенных разделителями друг от друга, порядок которых не важен.

Один из типичных примеров использования перечислений изображен на рисунке **??**.

Правильными ответами будут строки:

- bool sign; unsigned int enom; unsigned int denom;
- bool sign; unsigned int denom; unsigned int enom;
- unsigned int enom; unsigned int denom; bool sign;
- unsigned int enom; bool sign; unsigned int denom;
- unsigned int denom; unsigned int enom; bool sign;
- unsigned int denom; bool sign; unsigned int enom;

Количество вариантов перестановок элементов в перечислении определяется, факториалом от числа его элементов. При наличии в ответе нескольких перечислений количество вариантов правильных ответов будет равно произведению чисел порядков каждого перечисления в отдельности.

При рассмотрении данного примера становиться ясно, что поддержка перечислений необходима.

1.2 Обзор существующих решений

Для осуществления анализа состояния поддержки перечислений, а так же обзора существующих решений, необходимо ввести термин

перечисление. Перечислением называется последовательность элементов, состоящих из одной или нескольких лексем, порядок элементов в ответе не важен, они могут отделены друг от друга лексемами-разделителями.

На данный момент существует множество типов вопросов для СДО Moodle. Типы вопросов по разному отрабатывают ответы студентов, некоторые используют регулярные выражения, другие находят наибольшие общие части эталонного ответа и ответа данного студентом, третьи строят синтаксические деревья для поиска ошибок. В данной главе будут проанализированы основные плагины типов вопросов Pmatch, Preg и Correct Writing.

1.2.1 Критерии анализа

Главным требованием к разрабатываемой системе является интеграция с СДО Moodle. Данное требование было выдвинуто заказчиком, его актуальность объясняется использованием данной системы заказчиком.

Так же важным требованием является возможность внедрения поддержки перечислений в существующий модуль. Выбранные для анализа плагины, работают с ответом по разному, даже формат задания эталонного ответа различен. Из данного обстоятельства выплывает актуальность этого требования.

Довольно часто вопросы для тестов создаются обычными учителями, преподающих разные предметы, и имеющих различные уровни инженерного образования. Что требует возможности создания вопросов, а так же описаний перечислений в простой и понятной форме. То есть не требующих от учителя дополнительных познаний.

Таким образом, необходимо сравнить существующие системы по таким критериям, как возможность внедрения поддержки перечислений в существующий модуль, возможность интеграции в СДО Moodle, возможность создания вопросов и описаний перечислений в простой и понятной форме.

1.2.2 Тип вопроса PMatch

Данный плагин разрабатывается сообществом Open University в рамках разработки СДО Moodle. Плагин позволяет проводить анализ ответа студента на основе шаблона заданного преподавателем. Шаблоны используемые плагином схожи с регулярными выражениями. Так же в плагине реализована проверка опечаток, проверка ответа с неполным совпадением.

В плагине присутствует частичная возможность описания перечислений с помощью альтернатив [1]. Однако этот метод не лишен недостатков:

– при использовании альтернатив, есть возможность указать варианты лексем, для определенной позиции, но не возможно описать зависимости между лексемами. Что приводит к тому что при описании перечислений, будут порождаться неправильные ответы;

– так же при использовании шаблонов в связке с перечислениями, сложность описания правильного ответа будет возрастать, а уровень наглядности, будет уменьшаться.

Не смотря на то, что система не подходит для решения поставленной задачи, так как в ней не возможно реализовать описание и обработка перечислений, она подходит для решения других задач. В частности, на текущий момент, данная система используется для обучения различным предметам и обработки результата ответа на естественном языке, где перечисления отсутствуют.

1.2.3 Тип вопроса Preg

Данный плагин разрабатывается на кафедре "Программное обеспечение автоматизированных систем" Волгоградского государственного технического университета, под руководством к.т.н. Сычева О.А. Плагин производит анализ ответа студента на основе

заданного преподавателем регулярного выражения. Так же плагин позволяет выводить подсказки о следующем символе, лексеме. Так же преподавателю предоставлена возможность просмотра графа построенного на основе заданного выражения, его синтаксического дерева и словесного описания. Более того есть возможность проверки регулярного выражения вводом примеров ответов [2].

В плагине присутствует возможность описания перечислений на основе использования альтернатив. Регулярные выражения во многом схожи с шаблонами, но имеют под собой четкую и проработанную математическую базу. Данный подход имеет общие недостатки с шаблонным подходом. Кроме этого данный плагин сложнее в использовании и понимании, так как вопросы строятся с использованием математической системы и представления ответа в виде графа. То есть плагин требует специфических знаний.

Так же как и в предыдущей системе подход к обработке перечислений изменить невозможно. Следовательно система не подходит для решения данной задачи.

1.2.4 Тип вопроса Correct Writing

Плагин Correct Writing также является разработкой кафедры "Программное обеспечение автоматизированных систем"Волгоградского государственного технического университета, под руководством Сычева О.А. Плагин основан на синтаксическом анализе ответов преподавателя и студента. Плагин позволяет выделять лексические и синтаксические ошибки студента. А также демонстрировать объяснение ошибки студенту в двух вариантах: в формате текстового сообщения или графического изображения [3, 4].

Плагин не позволяет описывать перечисления. Единственный способ описать ответ содержащий перечисления - это полный перебор порядков элементов перечисления, что влечет за собой необходимость ввода описаний каждой лексемы для каждого варианта эталонного ответа.

Однако с другой стороны, данный плагин не использует специфических описаний, эталонный ответ, записывается как есть. Что позволяет говорить о том, что данный плагин прост и понятен для преподавателей, вне зависимости от предмета который они преподают и специфики их знаний.

Еще одним положительным моментом является возможность внедрения в плагин поддержки перечислений.

1.3 Анализ полученных результатов

Перед тем, как проводить анализ полученных результатов при помощи метода анализа иерархий, проведем первичный анализ результатов.

Все анализируемые системы (PMatch, Preg, CorrectWriting) являются плагинами для СДО Moodle, следовательно, из процесса сравнения данный критерий исключается как выполняющийся для всех систем.

Сформируем матрицу весов критериев. Для этого, проранжируем данные критерии по важности.

В процессе анализа было выявлено два критерия – возможность внедрения перечислений, возможность создания вопроса в простой и понятной форме. Определим их относительную важность. Первый критерий имеет существенную значимость над вторым. Исходя из ранжирования критериев Саати [5], можно составить матрицу, строками которой будут последовательности (1, 6), (0.165, 1). Вычисляя собственные значения данной матрицы, мы получим значения весов равные 2.45, 0.408. Нормализуя данный вектор, получаем значения весов равные 0.857, 0.143.

Проранжируем значения критериев. Для всех критериев возможно установить два значения – отсутствует, с численным значением «0», и присутствует, с численным значением «1».

Из таблицы 1 видно, что наилучшим вариантом является CorrectWriting. Так как удовлетворяет приведенным критериям.

Другие варианты, такие Preg, а также PMatch, нельзя признать до конца применимыми к данной задаче, так как они не обеспечивают возможности внедрения поддержки перечислений.

Таблица 1 — Результаты анализа систем

Системы	Внедрение перечислений	Простота создания вопроса	Общая оценка	Нормализованная оценка
	0.857	0.143		
PMatch	0	0	0	0
Preg	0	1	0.143	0.125
CorrectWriting	1	1	1.0	0.875

Цель и задачи исследования

Целью данной работы является снижение трудоемкости создания вопросов типа Correct Writing для изучения формализованных языков, с помощью поддержки перечислений.

Для достижения поставленной раннее цели необходимо решить следующие задачи:

– исследовать спецификации языков C/C++, выбрать структуры языка, которые могут быть записаны в любом порядке, не изменяя работы программы;

– разработать алгоритм определения наилучшего порядка элементов перечисления;

– реализовать разработанный алгоритм и оценить качество его работы;

– интегрировать алгоритм в структуру плагина Correct Writing.

Исходя из этого, в первую очередь, необходимо выбрать те операторы языка, которые позволяют включать в себя перечисления.

2 Разработка модуля поддержки перечислений

2.1 Описание интеграционных особенностей модуля

Структуру модуля CorrectWritting наиболее точно отражает диаграмма компонентов, изображенная на рисунке 1 [6].

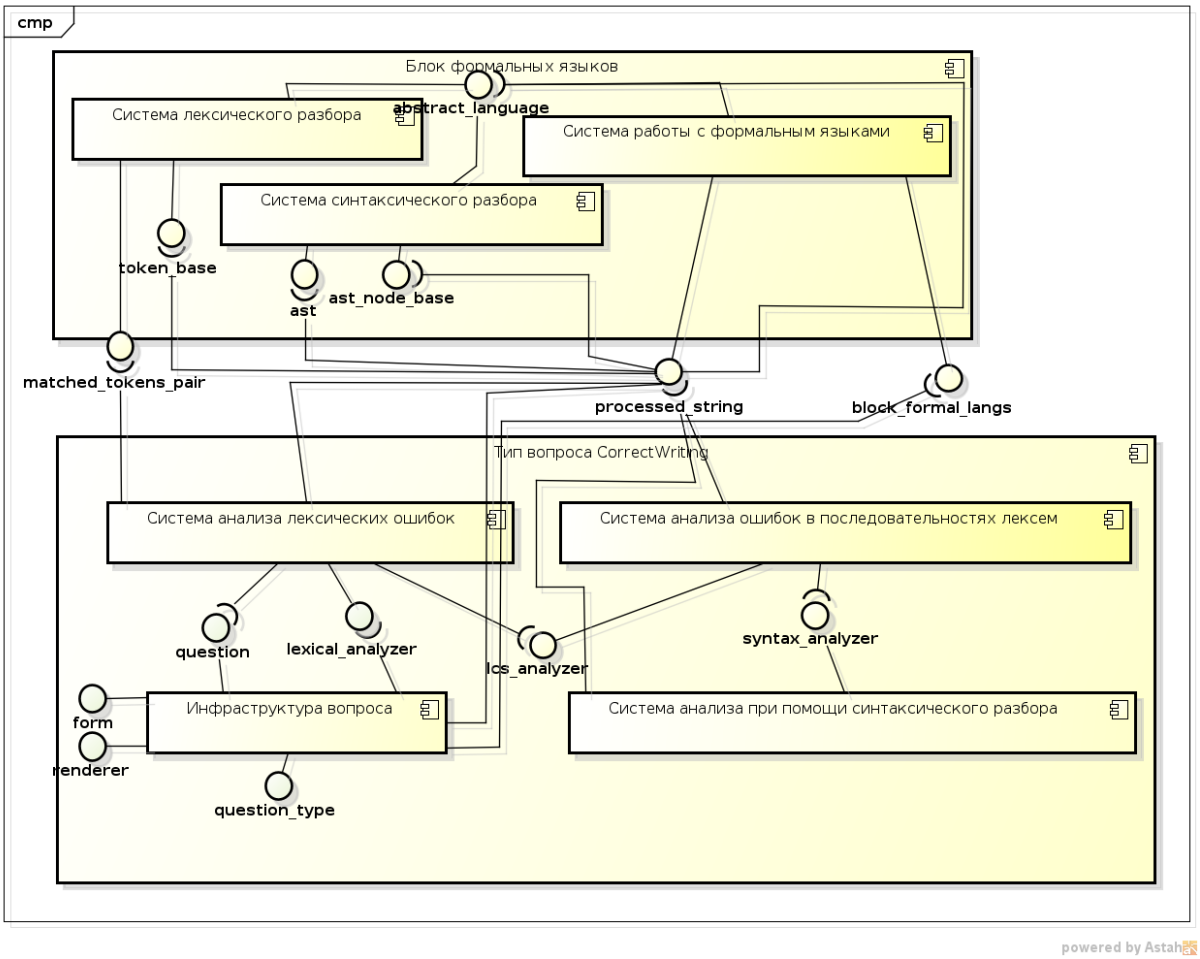

Рисунок 1 — Диаграмма компонентов модуля CorrectWriting

Процесс обработки ответа студента представлен на диаграмме потоков, рисунок 2 [7].

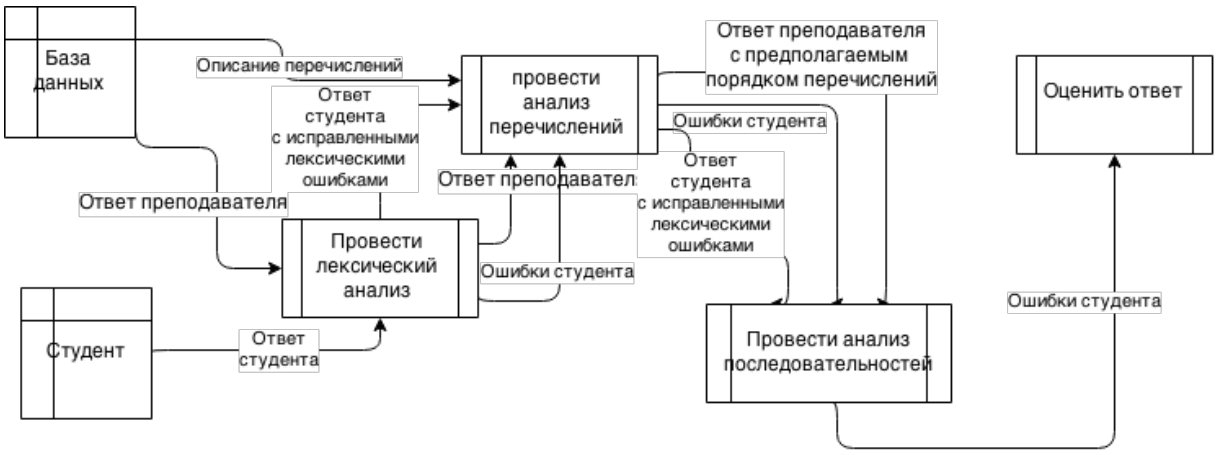

Рисунок 2 — Диаграмма потоков данных модуля CorrectWriting

Для каждого этапа анализа ответа, реализован собственный анализатор. Все анализаторы используют один и тот же набор входных и выходных данных. Набор входных данных представляет собой совокупность эталонного ответа, ответа студента, а также вектор найденных ранее ошибок. А набор выходных совокупность ошибок, найденных на этапе анализа, и в некоторых случаях исправленный ответ студента.

Исходя из структуры модуля было принято решение, разработать дополнительный анализатор определяющий порядок лексем в эталонном ответе, который позволяет получить наибольшую общую подпоследовательность двух ответов студента и эталонного. Дополнительным входным параметром будет описание перечислений соответствующие эталонному ответу. С другой стороны необходимо разработать средство для ввода описаний перечислений при создании нового вопроса. С этой целью было принято решение разработать модуль, определяющий возможные перечисления в эталонном ответе для языков C\C++, а также графический модуль позволяющий исправить найденные ранее описания, а также добавить новые, в специфических случаях, а также в случае создания вопроса с ответом на других языках.

2.2 Разработка модели представления перечислений

Описанием перечислений является вектор описаний элементов перечислений. Описанием элемента перечисления есть вектор двух чисел - номеров первой и последней лексем принадлежащих элементу.

После определения порядка внедрения, в существующую систему, встает вопрос о том какие данные необходимо хранить для работы модуля, а также вопрос о формате их хранения.

Так как в модуле, каждый из анализаторов, является отключаемым, необходимо разработать формат позволяющий обеспечить безаварийную работу, в случае отключения разрабатываемого модуля.

Наиболее точно описать перечисления можно описав границы его элементов. Описанием элемента перечислений служит пара индексов первой и последней лексем принадлежащих элементу в эталонном ответе. Тогда вектор описаний элементов перечислений, будет служить его описанием.

В CorrectWriting используется специфический класс связывающий пару ответов, эталонный и студента. Так как анализатор перечислений изменяет копию правильного ответа, было принято решение расширить класс эталонным ответом с измененным порядком лексем.

Так же существует проблема вложенности перечислений. Суть ее заключается в том, что при наличии вложенных перечислений, при перестановке элементов перечисления-контейнера, описания вложенного перечисления необходимо обновлять.

2.3 Исследование стандартов языка C\C++

Для разработки модуля определения перечислений для языков C\C++, необходимо выделить правила выделения перечислений. С целью выделения таких правил было проведено исследование стандартов языка C\C++ [8, 9].

В ходе анализа были выделены правила описанные в таблице 2:

Таблица 2 — Результаты исследования стандартов языка C\C++

Операция	Правило
Сложение	Оба аргумента данной операции являются являются элементами перечисления. В случае нескольких подряд идущих операций одного типа, все их аргументы являются элементами перечисления.
Умножение	
Конъюнкция	
Дизъюнкция	
Эквивалентность	
Не эквивалентность	
Битовое "И"	
Битовое "ИЛИ"	
Битовое исключающее "ИЛИ"	
Деление	Для нескольких подряд идущих операций одного типа, все их аргументы, кроме первого, являются элементами перечисления.
Вычитание	
Целочисленное деление	
Присваивание	Для нескольких подряд идущих операций присваивания, все их аргументы, кроме последнего, становятся элементами перечисления.
Объявление перечисляемого типа	Все объявленные перечислители являются элементами перечисления.
Объявление переменных одного типа	Все объявленные переменные, в том числе указатели, включая инициализацию, при ее наличии, являются элементами перечисления.
Объявление классов	Все уровни доступа и слоты являются элементами перечисления. Все поля, прототипы функций, функции с реализацией одного уровня доступа являются элементами перечисления.

Продолжение таблицы 2

Операция	Правило
Объявление структур	Все объявленные поля структур являются элементами перечисления.
Объявление объединений	

Для определения перечислений в сложных случаях, когда автоматически это сделать не удается, таких как объявление переменных разных типов, объявления с присваиванием объявленной ранее переменной, было принято решение разработать графический интерфейс редактирование перечислений. Требования к графическому интерфейсу, а также макеты представлены в приложении А.

2.4 Разработка алгоритмов решения задач

Для решения задачи обработки перечислений был разработан следующий алгоритм работы модуля:

а) проверка входных данных, то есть описания перечислений;
б) поиск порядка изменения перечислений с учетом вложенности;
в) поиск возможных порядков перечислений;
г) выбор наиболее подходящих порядков.

2.4.1 Алгоритм проверки входных данных

Для разработки данного алгоритма, для начала необходимо определить критерии оценки корректности входных данных. Так как входными данными является описание перечислений в эталонном

ответе, необходимо определить критерии корректности описаний. Этими критериями являются:

– отсутствие пересечений элементов перечислений;

– вложенность перечислений разрешена, но при условии что вложенное перечисление вложено только в один элемент другого или является разделителем только двух элементов;

– элементы перечислений не выходят за границы предложения.

Исходя из выше сказанного, алгоритм проверки входных данных, описывается следующей последовательностью шагов:

а) проверить отсутствие пересечений элементов, внутри перечислений, и между перечислениями;

б) проверить правильность вложенности перечислений;

в) проверить, не выходят ли перечисления за границы предложения.

Входными данными к данному алгоритму являются описания перечислений в эталонном ответе.

2.4.2 Алгоритм поиска порядка изменения перечислений с учетом вложенности

Необходимость данного алгоритма, объясняется тем, что при наличии вложенности, перестановки одного перечисления, влияют на позиции элементов другого. Из этого следует необходимость задания строгой очереди изменения порядков элементов перечислений. При этом вложенные перечисления необходимо обработать, раньше перечислений-хранилищ. Вложенность одного перечисления в другое можно определить из их описаний, по следующим правилам:

– если все элементы перечисления включены в границы одного элемента другого перечисления, то это перечисление вложено;

– если все элементы перечисления заключены между границами элементов другого, то это перечисление вложено;

Алгоритм поиска порядка изменения перечислений с учетом вложенности, в соответствии с определенными выше правилами, описан на рисунке 3.

Рисунок 3 — Алгоритм поиска порядка изменения перечислений с учетом вложенности

Входными данными к данному алгоритму являются описания перечислений в эталонном ответе.

2.4.3 Алгоритм поиска возможных порядков

Данный алгоритм используется для исключения полного перебора вариантов правильного ответа. Этот алгоритм сводиться к полному перебору только в двух ситуациях:

– если в предложении отсутствуют элементы перечислений;

– если в предложении для каждого перечисления элементы перечислений или отдельные лексемы им принадлежащие записаны в последовательности из которой можно получить все возможные порядки элементов перечисления.

Алгоритм поиска возможных порядков, описан на рисунке 4.

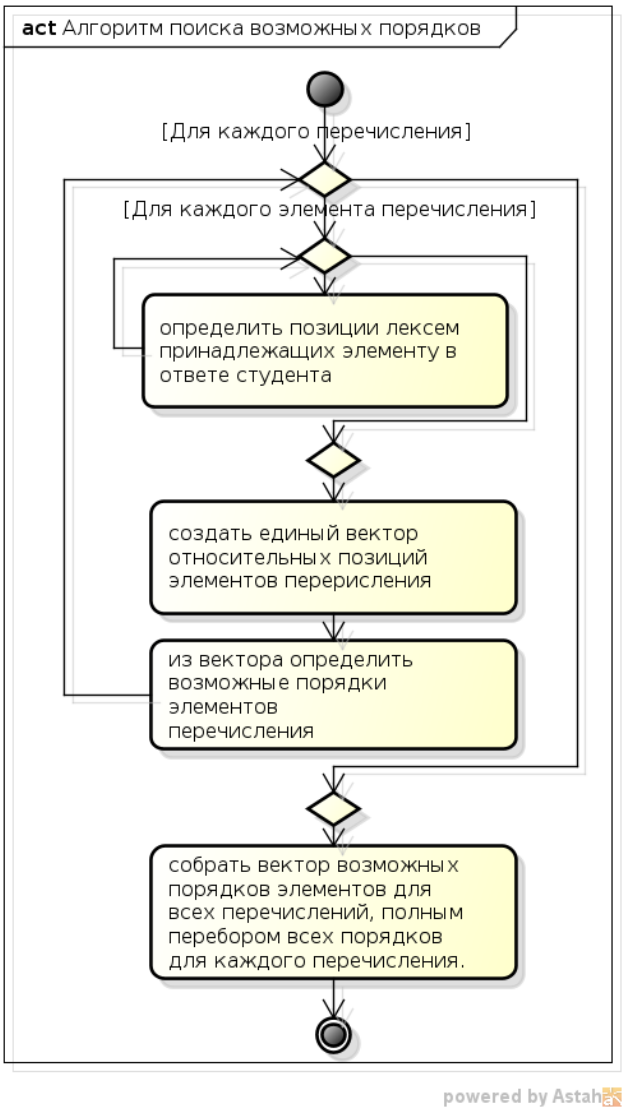

Рисунок 4 — Алгоритм поиска возможных порядков

Входными данными к данному алгоритму являются описания перечислений в эталонном ответе, эталонный ответ, ответ студента.

2.4.4 Алгоритм выбора наиболее подходящих порядков

Во-первых необходимо определить критерий выбора наиболее подходящих порядков. Так как мы сравниваем два ответа между собой, этим критерием является наибольшая общая часть двух ответов. Следовательно наиболее подходящие порядки, будут те порядки, использование которых порождает наибольшую общую часть двух ответов.

Алгоритм выбора наиболее подходящих порядков, описан на рисунке 5.

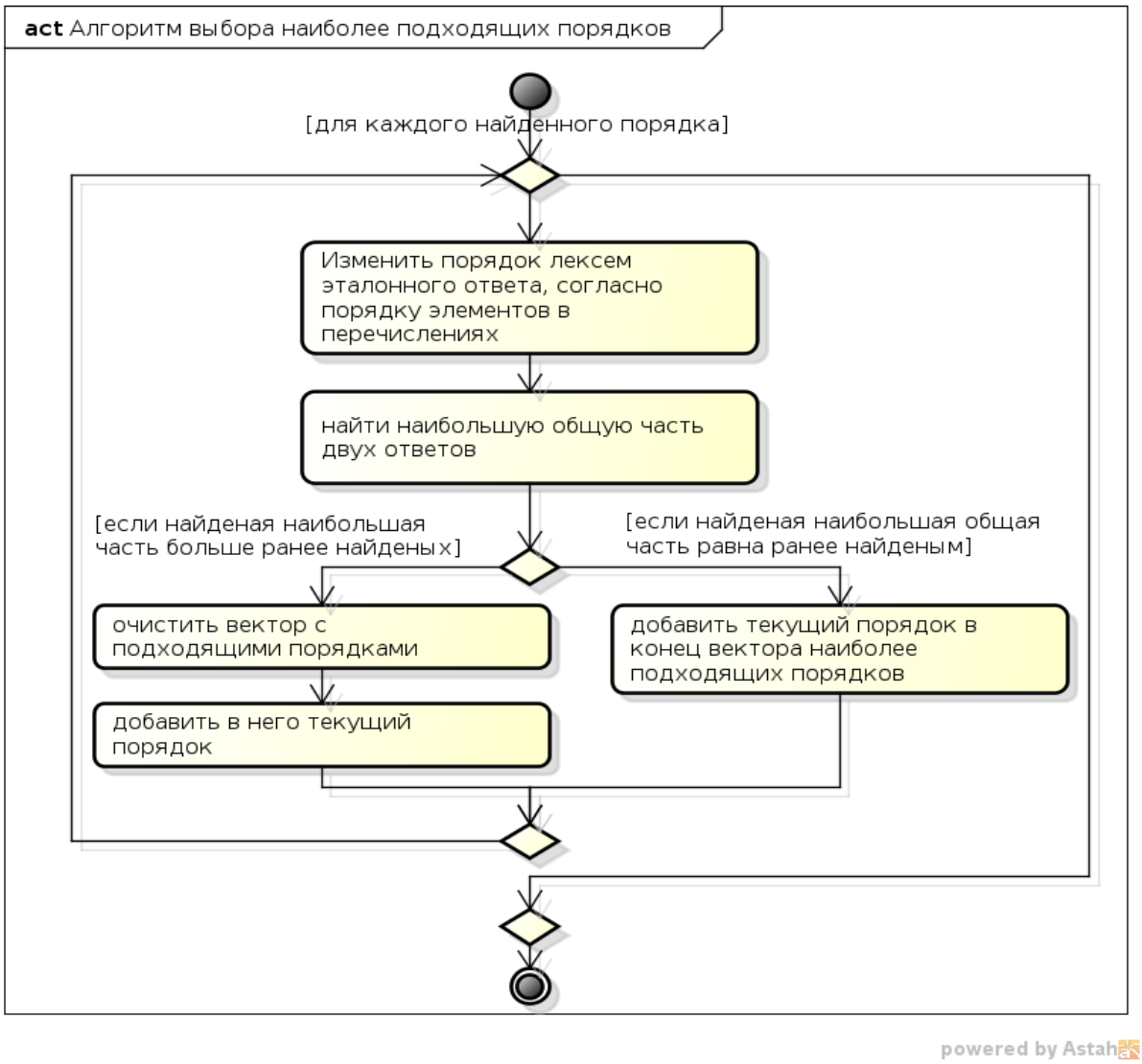

Рисунок 5 — Алгоритм выбора наиболее подходящих порядков

Входными данными к данному алгоритму являются описания перечислений в эталонном ответе, эталонный ответ, ответ студента.

2.5 Оценка принятого решения

Данный алгоритм сложнее использования полного перебора, сложность которого описана в формуле (1).

$$O(N*M*K), \quad (1)$$

где N - число вариантов правильных ответов;

M - число лексем правильного ответа;

K - число лексем в ответе студента.

Приведенный алгоритм имеет сложность описанную в форуле (2).

$$O(R*M*K*Z*X^2*Y*E*P), \quad (2)$$

где R - количество найденных вариантов правильного ответа $1 \leq R \leq N$,

Z - число лексем принадлежащих перечислениям в правильном ответе;

Y - число элементов в перечислении;

X - число перечислений;

E - число лексем элемента перечисления найденных в ответе студента;

P - число найденных порядков элементов перечисления.

С другой стороны, несмотря на возросшую сложность, алгоритм работает быстрее полного перебора, из-за более простых внутренних операций над ответами.

Еще одним плюсом данного решения является уменьшения трудоемкости создания новых вопросов. Это объясняется отказа от необходимости полного перебора, в пользу добавления описания перечисления для одного правильного ответа.

Выводы

Введение специального алгоритма позволит отказаться от полного перебора вариантов, за исключением сложных случаев, когда в ответе студента содержаться все варианты перестановки элементов для каждого перечисления. Так же процесс создания нового вопроса становиться прост и понятен.

3 Оценка разработанного решения

3.1 Описание процесса интеграции

Интеграция в модуль происходила в двух направлениях:

- интеграция в процесс создания ответа;
- интеграция в процесс оценивания ответа.

Интеграция в процесс создания ответа, заключается в добавлении к форме создания и редактирования вопроса окна редактирования перечислений для каждого эталонного ответа. А также реализации сериализации и десериализации описаний перечислений, для хранения в базе данных.

Интеграция в процесс оценивания ответа, заключается во включении в процесс нового анализатора, и реализации адекватного взаимодействия анализаторов.

3.2 Оценка структурной целостности решения

Модуль разрабатывался с использованием объектно-ориентированного подхода [10, 11].

Модуль соответствует требуемому сообществом разработчиков Moodle стилю написания кода для языков PHP и JavaScript [12, 13, 14].

Часть модуля связанная с работой на клиенте, написана с использованием библиотеки Yui [15], являющийся официальной библиотекой используемой в Moodle.

На рисунке 6, прадставлен фрагмент диаграммы классов типа вопроса CorrectWriting.

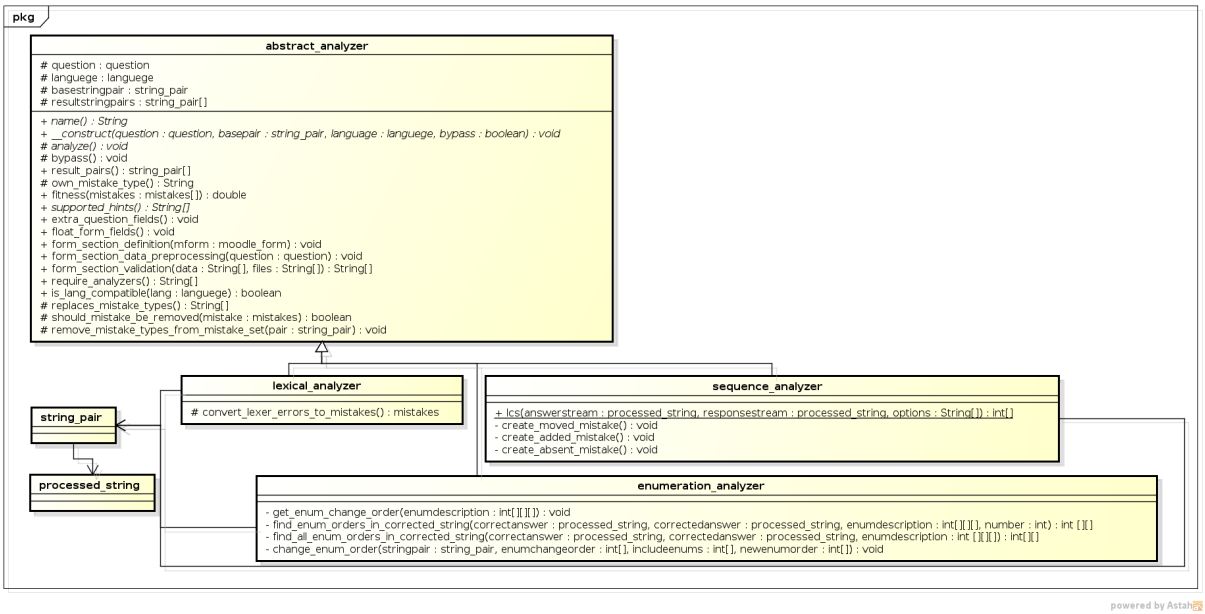

Рисунок 6 — Фрагмент диаграммы классов

Enumeration analyzer разработан как часть модуля. Его методы являются реализаций алгоритмов описанных во второй части данной пояснительной записки. Соответствие алгоритмов методом представлено в таблице 3.

Таблица 3 — Соответствие методов алгоритмам

Алгоритм	Метод
алгоритм поиска порядка изменения перечислений с учетом вложенности	get_enum_change_order
алгоритм поиска возможных порядков	find_enum_orders_in_corrected_ string и find_all_enum_orders_in_corrected_ string
алгоритм выбора наиболее подходящих порядков	change_enum_order и analyze

Метод find_enum_orders_in_corrected_string находит предполагаемые порядки перечисления в ответе студента.

Метод find_all_enum_orders_in_corrected_string, используя find_enum_orders_in_corrected_string, находит порядки для всех

перечислений и создает варианты порядка лексем для эталонного ответа.

Метод change_enum_order изменяет порядок лексем в эталонном ответе, а также обновляет описание перечислений.

Метод analyze является главным в модуле. Он объединяет работу всех модулей, и выбирает наиболее подходящие эталонные ответы.

На диаграмме последовательности, представленной на рисунке 7, описан процесс интеграции в процесс оценивания ответа студента.

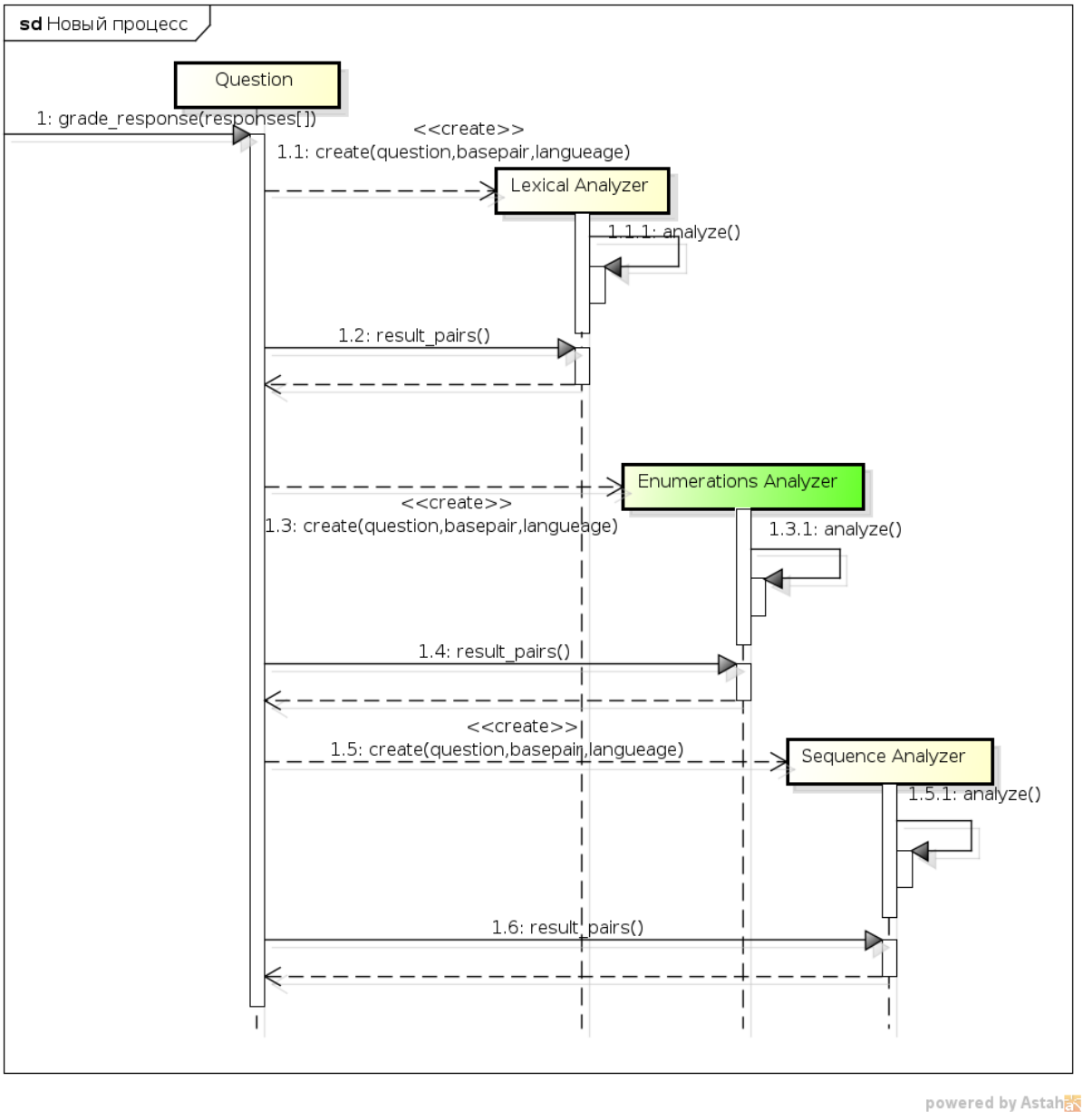

Рисунок 7 — Диаграмма последовательности

3.3 Оценка работы модуля на тестовом примере

Тестирование проведено на трех примерах:

а) тестирование работы графического интерфейса в приложении А;

б) тестирование работы анализатора на этапе оценивания ответа в приложении А;

в) тестирование скорости работы серверной части, по сравнению с CorrectWriting без разработанного модуля.

Модуль выдержал тестирование графического интерфейса и тестирование работы анализатора на этапе оценивания ответа.

Так же были разработаны модульные тесты:

а) для анализатора перечислений 36 тестов, на работу всех функций модуля;

б) для определения перечислений 29 тестов, на все правила;

в) интеграционное тестирование 15 тестов, на взаимодействие всех анализаторов.

Тестирование скорости работы серверной части, будет рассмотрено подробнее. В качестве эталонного ответа используется строка "t == a || b && c || k != true; содержащая четыре перечисления:

а) перечисление связанное с эквивалентностью, элементами которого являются:

1) лексема "t";
2) лексема "a";

б) перечисление связанное с логическим сложением, элементами которого являются:

1) лексема "b";
2) лексема "c";

в) перечисление связанное с не эквивалентностью, элементами которого являются:

1) лексема "k";
2) лексема "true";

г) перечисление связанное с логическим умножением, элементами которого являются:

1) перечисление связанное с эквивалентностью, то есть подстрока "t == a";

2) перечисление связанное с логическим сложением, то есть подстрока "b && c";

3) перечисление связанное с не эквивалентностью, то есть подстрока "k != true".

Для CorrectWriting используется полный перебор, количество вариантов равно произведению факториалов числа элементов каждого перечисления, в данном случае 2!*2!*2!*3!, что равно 48.

Временные замеры проведенные на сервере показали, что работа модуля увеличивает скорость работы CorrectWriting на порядок. Результаты замеров:

а) для CorrectWriting с полным перебором вариантом - 1.079 сек;

б) для CorrectWriting с использованием модуля - 0.045 сек.

Модуль прошел тестирование, и доказал свою работоспособность.

3.4 Оценка продуктивности внедрения модуля

После разработки модуля был проведен анализ продуктивности его использования. Были проанализированы тестовые вопросы проверяющие компетенцию студентов 1 и 2 курсов Волгоградского Государственного Технического Университета по дисциплинам "Программирование"и "Основы программирования". Анализу подверглись 290 вопросов, 40 из них содержали в эталонном ответе перечисления.

Перечисления в вопросах были следующих типов:

- перечисления связанные с арифметическими операциями;
- перечисления связанные с объявлением структур;
- перечисления связанные с объявлением переменных и указателей;

– перечисления связанные с составным типом объявляемой переменной.

Внедрение модуля позволило сократить количество необходимых для ввода эталонных ответов на 45 процентов. В вопросах представлены только перечисления с двумя элемента, поэтому процент меньше 50. В будущем ответы будут содержать более сложные перечисления из большего числа элементов.

3.5 Пркатическая значимость

Работа была представлена на двух смотрах:

а) смотр-конкурс научных, конструкторских и технических работ студентов ВолГТУ 2014 года;

б) смотр-конкурс научных, конструкторских и технических работ студентов ВолГТУ 2015 года.

По результатам выступлений были напечатаны тезисы.

Выводы

Разработанный и внедренный модуль соответствует парадигме объектно-ориентированного программирования, поддерживает и сочетается со структурой CorrectWriting. Более того практически доказана продуктивность внедрения, и снижение трудоемкости создания вопросов данного типа. Что то по поводу производительности.

Из проведенных выше оценок и исследований следует правильность принятого решения.

Заключение

В ходе данной работы были исследованы спецификации и стандарты языков C/C++. На основе этих спецификаций были выделены правила автоматического определения перечислений.

Был разработан алгоритм определения наилучшего порядка элементов перечисления и доказана его эффективность.

Данный алгоритм был разработан и протестирован, так же была проведена оценка его эффективности. Так же была проанализирована степень уменьшения трудоемкости создания вопросов на основе существующего банка вопросов кафедры программного обеспечения автоматизированных систем Волгоградского Государственного Технического Университета.

Так же была проведена интеграция модуля в тип вопроса CorrectWriting, проверено его взаимодействие с другими элементами существующей системы.

Таким образом каждая из поставленных задач была решена. Что позволило достичь поставленной цели.

Список использованных источников

1 Pmatch question type [Electronic resource]. – Mode of access : https://moodle.org/plugins/view.php?plugin=qtype_pmatch (date of access 11.03.2015).

2 Сычев, О.А. Архитектура программного обеспечения тестового вопроса PREG с оценкой ответа по регулярным выражениям, поддерживающего возможность подсказок продолжения совпадения / О. А. Сычев, В. О. Стрельцов, Д. В. Колесов // Известия ВолгГТУ : межвуз. сб. науч. ст. / ВолгГТУ. – Волгоград, 2012. – №15 – С. 99-104.22

3 Мамонтов, Д.П. Автоматизированная генерация описаний ошибок на основе правил атрибутивной грамматики в модуле типа вопроса Correctwriting / Мамонтов Д.П., Сычев О.А. // Тезисы докладов смотра-конкурса научных, конструкторских и технологических работ студентов Волгоградского государственного технического университета, Волгоград, май 2014 г. / редкол. : А.В. Навроцкий (отв. ред.) [и др.] ; ВолгГТУ, СНТО. - Волгоград, 2014. - С. 149-150.

4 Мамонтов, Д. П. Автоматизация поиска ошибок в ответе студента на тестовый вопрос с помощью редакционных расстояний / Мамонтов Д. П. // Всероссийский конкурс научно-исследовательских работ студентов и аспирантов в области информатики и информационных технологий : сб. науч. работ. В 3 т. Т. 3 / ФГАОУ ВПО «Белгородский гос. нац. исслед. ун-т». - Белгород, 2012. - С. 102-106.

5 Саати Т, Принятие решений: Метод анализа иерархий / Т. Саати; перевод Р. Г. Вачнадзе; Радио и связь.- Москва, 1993. - 278 с.

6 Unified Modeling Language [Electronic resource]. – Mode of access : http://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF/ (date of access 12.03.2015).

7 Data Flow Diagram [Electronic resource]. - Mode of access : http://www.smartdraw.com/data-flow-diagram/ (date of access 12.03.2015).

8 C++ Language Reference [Electronic resource]. - Mode of access : https://msdn.microsoft.com/en-us/library/3bstk3k5.aspx (date of access 12.03.2015).

9 The C++ Programming Language [Electronic resource]. - Mode of access : http://www.stroustrup.com/C++.html (date of access 12.03.2015).

10 Мартин, Р. Чистый код: создание, анализ и рефакторинг. Библиотека программиста. / Р. Мартин; перевод Е. Матвеев; Питер.- СПб, 2010.- 464 с.

11 Макконнелл, С. Совершенный код. Мастер-класс / С. Макконнелл; Русская редакция. - М, 2010.- 896 с.

12 Coding style [Electronic resource]. - Mode of access : https://docs.moodle.org/dev/Coding_style (date of access 12.03.2015).

13 JavaScript guidelines [Electronic resource]. - Mode of access : https://docs.moodle.org/dev/JavaScript_guidelines (date of access 12.03.2015).

14 Javascript/Coding style [Electronic resource]. - Mode of access : https://docs.moodle.org/dev/Javascript/Coding_style (date of access 12.03.2015).

15 YUI API Docs [Electronic resource]. - Mode of access : http://yuilibrary.com/yui/docs/api/ (date of access 12.03.2015).

Приложение А
Техническое задание