Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техника

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись
« ____» _____ 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – «Информатика и вычислительная техника»

Генератор тестовых заданий по программированию

Руководитель _____ ст. преподаватель К. В. Пушкарев
подпись, дата

Выпускник _____ М. В. Гильденберг
подпись, дата

Нормоконтролер _____ К. В. Пушкарев
подпись, дата

Красноярск 2021

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техники

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись
« ___ » _____ 20 ___ г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Гильденбергу Максиму Владимировичу.

Группа: КИ17-08Б. Направление (специальность): 09.03.01 «Информатика и вычислительная техника».

Тема выпускной квалификационной работы: «Генератор тестовых заданий по программированию».

Утверждена приказом по университету № 5631/с от 27.04.2021.

Руководитель ВКР: Пушкарев К. В., старший преподаватель кафедры ВТ ИКИТ СФУ.

Исходные данные для ВКР: нет.

Перечень разделов ВКР:

1. Анализ задания на выпускную квалификационную работу.
2. Проектирование и реализация системы.
3. Инструкции.

Перечень графического материала: не требуется.

1 Задание

Разработать приложение для персональных компьютеров — генератор тестовых заданий по программированию на языке C++.

2 Основные требования

Разработанное приложение должно быть кроссплатформенным для настольных систем (ОС Windows, Linux), иметь графический или консольный пользовательский интерфейс, открытый исходный код и обладать следующими возможностями:

а) генерация случайных вопросов на основе задаваемых пользователем шаблонов вопросов, содержащих постоянные данные, места для подстановки случайных параметров, управляющие конструкции и т. п.;

б) генерация вопросов типа небольших задач, сопровождаемых фрагментами программного кода на языке C++;

в) поддержка вопросов со следующими типами ответов: 1) множественный выбор с одним и несколькими правильными ответами; 2) короткий текстовый ответ; 3) числовой ответ; 4) да/нет; 5) на соответствие;

г) поддержка вывода тестовых заданий в файловом формате Moodle XML;

д) задание конфигурации вопросов в текстовом формате, удобном для человека;

е) генерация примеров как заведомо корректного, так и некорректного кода с синтаксическими и логическими ошибками.

Руководитель ВКР _____ К. В. Пушкарев
подпись

Задание принял к исполнению _____ М. В. Гильденберг
подпись

31 октября 2020 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Генератор тестовых заданий по программированию» содержит 29 страниц текстового документа, 10 иллюстраций и 7 использованных источников.

ГЕНЕРАТОР ТЕСТОВЫХ ЗАДАНИЙ, ТЕСТИРОВАНИЕ, ПРОГРАММИРОВАНИЕ НА C++, PYTHON, HTML, MOODLE XML.

Цель работы: разработать приложение для персональных компьютеров — генератор тестовых заданий по программированию на языке C++.

Для достижения цели в работе решаются следующие задачи:

- анализ задания на выпускную квалификационную работу;
- проектирование;
- реализация приложения.

Когда преподаватель составляет тестовые задания для проверки знаний студентов, то это занимает большое количество времени и поэтому необходима программа, которая способна генерировать эти тестовые задания с использованием шаблонов и гибкой настройкой для составителя тестов.

Во введении раскрывается актуальность работы, указываются цели и задачи.

В первой главе проводится анализ задания на существующие решения, а также выбор инструментов реализации приложения.

Во второй главе описывается проектирование и реализация приложения, сохранение в формате Moodle XML, а также алгоритм генерации тестовых заданий.

В третьей главе прописаны инструкции для пользователя и разработчика, а также описание работы с конфигурационным файлом и директориями.

В результате работы создано приложение для генерации тестовых заданий по программированию на C++.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3
1 Анализ задания на выпускную квалификационную работу 4
1.1 Обзор существующих решений 4
1.1.1 LMS Moodle 4
1.1.2 Computational Quiz Generation 5
1.1.3 Автоматическая генерация тестовых заданий для языка программирования C 6
1.2 Интерфейс программы 8
1.3 Диаграмма прецедентов 10
1.4 Выбор инструментов 11
1.5 Итоги анализа задания 12
2 Проектирование и реализация приложения 12
2.1 Структура приложения 12
2.2 Графический интерфейс 15
2.3 Сохранение и загрузка Moodle XML 18
2.4 Алгоритм генерации тестовых заданий 21
2.4.1 Вопрос со сгенерированной синтаксической ошибкой 21
2.4.2 Вопрос со сгенерированной логической ошибкой 21
2.4.3 Вопрос, при каком входном значении программа выдаст соответствующий результат после выполнения кода 21
2.4.4 Вычисление результата выполнения показанного кода 22
2.4.5 Вопрос Да/Нет 22
2.4.6 Описание классов 22
3 Инструкции 23
3.1 Инструкция пользователя 23
3.2 Инструкция разработчика 27
ЗАКЛЮЧЕНИЕ 28
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29

ВВЕДЕНИЕ

В процессе обучения студенты на первых курсах изучают основы программирования. Как правило, одним из языков программирования для изучения является C++ и иногда преподавателям приходится составлять тестовые задания для проверки и оценки знаний студентов. Для того, чтобы создавать вопросы для тестирования вручную, требуется достаточно большое количество времени. В связи с этим необходима программа, которая способна генерировать эти тестовые задания с гибкой настройкой для составителя тестов.

Цель работы: разработать приложение для персональных компьютеров — генератор тестовых заданий по программированию на языке C++.

Приложение должно быть кроссплатформенным, поддерживать вывод тестовых заданий в формате Moodle XML, а также показ предварительного просмотра теста.

Приложение должно обладать следующими возможностями:

а) генерация случайных вопросов на основе прототипов, заданных схемами и правилами;

б) генерация вопросов типа небольших задач, сопровождаемых фрагментами программного кода на языке C++;

в) поддержка вопросов со следующими типами ответов: множественный выбор с одним и несколькими правильными ответами; короткий текстовый ответ; числовой ответ; «Да/Нет»; на соответствие;

г) поддержка вывода тестовых заданий в файловом формате Moodle XML;

д) задание схем и правил в текстовом формате, удобном для человека;

е) генерация примеров как заведомо корректного, так и некорректного кода с синтаксическими и логическими ошибками.

Для достижения цели в работе решаются следующие задачи:

- анализ задания на выпускную квалификационную работу;

- проектирование;

- реализация приложения.

1 Анализ задания на выпускную квалификационную работу

Каждый год информационные технологии развиваются, в том числе и в образовании. Студентам становится проще изучать предметы с помощью дистанционного обучения. Так многие вузы используют систему Moodle для создания собственных курсов, в которых удобнее изучать лекционный материал и проходить тестирования для проверки и оценки знаний студентов. Многие преподаватели создают тесты вручную, на что уходит достаточно большое количество времени. Поэтому необходимо упростить вариант создания тестов, в чем отлично поможет генерация тестов. Чтобы студенты не привыкали к уже сгенерированным тестам, а набивали руку, выполняя все новые задания после каждой попытки, необходимо генерировать тесты случайным образом.

Данная работа посвящена реализации метода генерации случайных вариантов тестовых заданий и вопросов по программированию.

1.1 Обзор существующих решений

1.1.1 LMS Moodle

LMS Moodle – электронная система для управления электронным образованием [1]. У Moodle есть встроенный редактор, позволяющий создавать лекции, опросы, задания и тесты. В ней прекрасно реализована тестовая система для базовой оценки знаний и прохождение курсов. Тестовые задания имеют функцию случайной генерации из составленной базы вопросов. Вопросы имеют письменную или графическую форму. Имеется возможность составить вопрос с предоставленным кодом на выбранном языке программирования. Ответы могут включать математические выражения с использованием переменных. Например, текст вопроса может выглядеть так: «Вычислить {a} + {b}», а правильный ответ можно записать в виде формулы «{a} + {b}. Все эти функции доступны при создании составителю тестов.

Генерация вопросов производится с использованием функций случайной генерации. Для каждого типа вопроса есть свои ответы, и они тоже перемешиваются при каждой генерации, чтобы не было повторений при прохождении нового теста. Проверка тестов выполняется из условий, в которых правильный ответ дает соответствующее количество баллов. По итогу теста все баллы суммируются и выводится результат выполнения для студента.

1.1.2 Computational Quiz Generation

CQG – это система генерации онлайн-тестов, реализованная с помощью технологии HTML-форм и ориентированная на оценку практических навыков [2]. В ней двадцать три теста, в которых основное внимание уделено трассировке кода на C, C++, Java и Python, процедурам, обычно преподаваемым на курсах компьютерных сетей, таким как вычисление кода Хэмминга, а также классическим алгоритмам шифрования и дешифрования. Фреймворк включает в себя код и детализированные процессы разработки новых вопросов и новых типов вопросов. Каждый вопрос представлен в виде шаблона. Студент предоставляет ответы в HTML-элементах ввода, таких как текстовые поля и раскрывающиеся списки, и нажимает кнопку «Проверить». CQG проверяет заполненный шаблон и выдает сообщение: «правильно» или «неправильно».

Процесс разработки типа вопроса основан на эталонной архитектуре и процедуре проверки, направленной на минимизацию сбоев сервера и ошибок оценки. CQG занимает нишу между тестовыми вопросами со множественным выбором и полноценными лабораторными работами или их симуляцией. При таком подходе можно выделить четыре основных шага:

а) преподаватель выбирает вычисление;

б) преподаватель создает вопрос, в шаблонной форме, путем замены одного или нескольких элементов заполнителями;

в) студент заполняет в шаблоне пропуски в полях ввода;

г) приложение автоматически проверяет решение студента, выполняя вычисления по заполненному шаблону.

В системе CQG тест состоит из списка вопросов. Каждый вопрос имеет свой тип вопроса. На сегодняшний день разработано 14 типов вопросов. CQG тесты эффективны для оценки практических навыков в активном чтении кода, алгоритмах шифрования/дешифрования.

Генерация вопросов основана на шаблонах. Код C разделен на глобальный код и основной код, чтобы упростить создание дополнительного кода, который иногда требуется для обработки проверки ответа. За шаблонами кода следуют шаблоны для аргументов командной строки, стандартного ввода и стандартного вывода. Каждый из шаблонов может содержать смесь простого текста и маркеров для текстового поля в заданном вопросе. На рисунке 1 используются маркеры $x0, $x1 и $y0; маркером может быть любая строка, которая еще не встречается в шаблонах.

Листинг 1 – Шаблон вопроса для ввода-вывода

```
Global code
#include <stdio.h>
Main code
int a,b;
scanf("%d",&a);
scanf("%d",&b);
if (a < b)
printf("Minimum: %dnn", a);
else
printf("Minimum: %dnn", a);
Command-line arguments
Standard input
$x0
$x1
Standard output
Minimum: $y0
Tuple list
[1,2,None]
[9,3,None]
[None,None,1]
[None,None,20]
[None,None,-2]
```

Помимо быстрой генерации вопросов и достаточно быстрой разработки типов вопросов, было три взаимозависимых цели проектирования CQG:

а) минимален по дизайну. Не используются клипы, JPEG изображения или анимации;

б) низкая нагрузка на сервер. В коде Хэмминга проверка ответов выполняется очень быстро для коротких битовых строк, подходящих для вопросов тестирования;

в) безопасность. В обоих типах вопросов ограничения проверяются на стороне сервера, чтобы защитить от взлома параметры HTML GET / POST.

1.1.3 Автоматическая генерация тестовых заданий для языка программирования C

Описанное в [3] приложение Java дает возможность автоматического создания тестов для языка программирования C. Созданный тест будет

проверять знания студента о способах определения функций на языке С. Студенту предлагается сгенерированный случайным образом вопрос. Правильность ответа студента анализируется с помощью регулярных выражений.

В заголовке функции С есть три поля:

а) тип возвращаемого результата;

б) имя функции;

в) список параметров функции (их тип и их название).

В данном приложении генерируются заголовки для функций, которые имеют 0, 1 или 2 параметра. Автоматическое построение заголовков функций для тестов основано на генерации случайных чисел. Чтобы установить тип возвращаемого результата из функции генерируется целое случайное число из списка {0,1,2,3}, где хранятся 4 элемента. Таким же образом количество параметров выбирается случайным образом из списка {0, 1, 2}. Для каждого параметра генерируются два других случайных числа: первое случайное число из списка {0, 1, 2}, определяющее тип данных параметра и второе случайное число из списка {0, 1}, определяющее, является ли параметр входным или выходным. Общее количество различных функций, которые можно автоматически сгенерировать, составляет 172.

Правильность ответа студента на вопрос, сгенерированный программой, проверяется с помощью регулярных выражений. Регулярные выражения используются для поиска или редактирования текста, и с помощью регулярного выражения определяется шаблон, который ищем. Они построены по определенным правилам. Чтобы упростить поиск, регулярные выражения определяются с использованием символов, метасимволов и квантификаторов.

Пример символов:

а) «.» – соответствует любому персонажу.

Пример метасимволов:

а) «\d» – соответствует любой цифре.

б) «\s» – соответствует пробелу.

Пример квантификаторов:

а) «+» – встречается 1 или более раз.

б) «*» – встречается 1 или более раз.

в) «?» – встречается 0 или 1 раз {n}.

г) «{n}» – встречается n раз.

Например, простейшее регулярное выражение для вопроса: «Функция, которая не возвращает никакого результата и не имеет параметров?».

Правильный ответ: «void f ()».

Поскольку в нем может быть больше пробелов, а ответ все еще правильный, регулярное выражение выглядит следующим образом: «\\s*void\\s+f\\s*\\(\\s*\\)\\s*».

1.2 Интерфейс программы

На основе задания на ВКР был сделан макет интерфейса (рисунки 1-2), который должен быть в конечном приложении. После запуска программы открывается главное окно, в котором указывается количество вопросов и нажимаем кнопку «Сгенерировать тест». Переходим в окно предварительного просмотра сгенерированного теста. В нем мы можем уже предварительно ответить на сгенерированные вопросы и после ответа на все вопросы всплывает уведомление и результате тестирования. Тест генерируется и сохраняется файл в формате HTML и XML в директории проекта output.

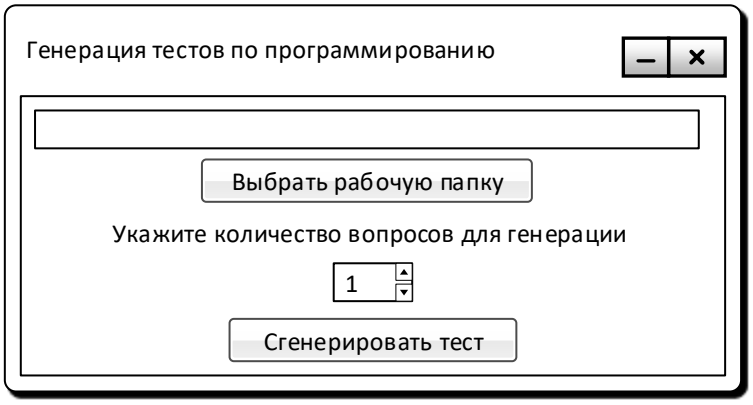

Рисунок 1 – Макет интерфейса главного окна

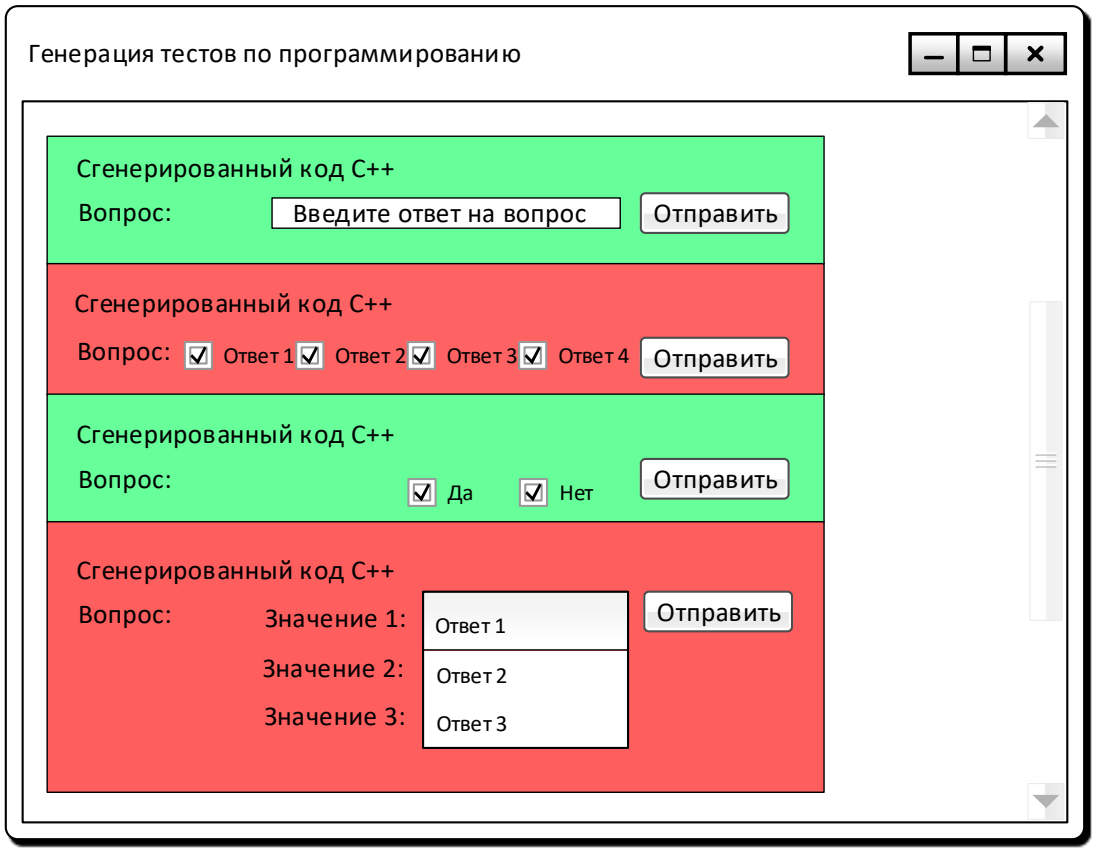

Рисунок 2 – Макет интерфейса предварительного просмотра теста

1.3 Диаграмма прецедентов

На рисунке 3 изображена диаграмма прецедентов.

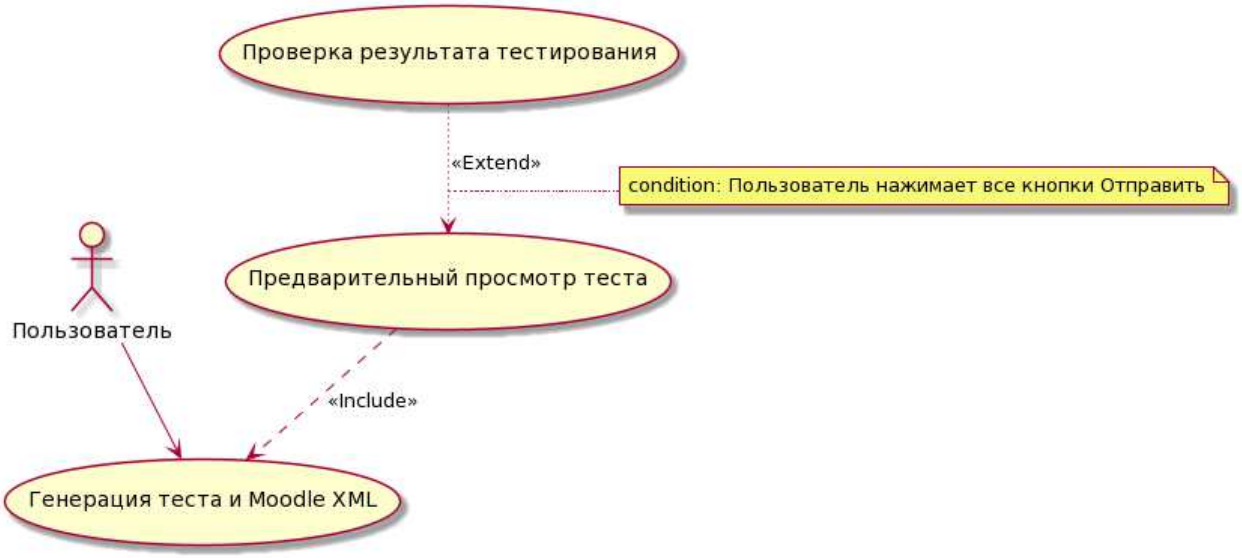

Рисунок 3 – Диаграмма прецедентов

Текстовое описание прецедентов приведено ниже.

Название прецедента: «Генерация теста и Moodle XML».
Действующее лицо: Пользователь.
Предусловие: Пользователь должен запустить программу
Главная последовательность:
1. Указать директорию рабочего каталога с конфигурационным файлом.
2. Выбрать количество вопросов в заданном диапазоне из конфигурационного файла.
3. Нажать кнопку «Сгенерировать тест».
4. Переход к предварительному просмотру теста.
5. Перейти в директорию основной программы, далее в директорию output.

Название прецедента: «Предварительный просмотр теста».
Действующее лицо: Пользователь.
Предусловие: Пользователь должен сгенерировать тест.
Главная последовательность:
1. Нажать кнопку «Сгенерировать тест» из главного окна.
2. Переход к предварительному просмотру теста.

Название прецедента: «Проверка результата теста».
Действующее лицо: Пользователь.
Предусловие: Тест должен быть сгенерирован.

Главная последовательность:

1. Ввести ответ в поле, либо выбрать из множественной выборки, либо сопоставить ответы.

2. Нажать кнопку «Отправить» под вопросом.

3. Результат отображается в уведомлении пользователю.

1.4 Выбор инструментов

Приложение должно работать на платформах Windows и Linux и обеспечивать графический интерфейс пользователя, т.е. приложение должно быть кроссплатформенным. Выбор стоял между следующими языками и фреймворками:

- Java с библиотекой Swing [4];
- C++ с библиотекой Qt [5];
- Python с библиотекой tkinter [6].

Java – это язык объектно-ориентированного программирования. Swing создает оконные приложения.

C++ – это компилируемый язык программирования со статической типизацией. C++ включает в себя поддержку объектно-ориентированного программирования.

Qt включает графический фреймворк, с помощью которого можно реализовать интерфейс программы. Также он включает в себя библиотеки для работы с XML, чтоб сгенерировать тест.

Python – это высокоуровневый язык программирования с динамической типизацией. В Python синтаксис обладает простотой, что делает написанный код легко читаемым.

Python предоставляет широкий выбор библиотек для генерации случайных значений или работы со списком. В нем имеются простые функции для реализации данной задачи со случайной генерацией. Так как реализация интерфейса не является первостепенной задачей, то библиотека tkinter прекрасно подойдет для создания простого оконного приложения с предварительным просмотром. В итоге, так как Python является одним из наиболее простых в использовании языков для быстрого прототипирования, то был выбран Python.

1.5 Итоги анализа задания

Проведя анализ аналогов, можно сделать вывод, что они имеют различное количество функций, выполняющих те или иные задачи, но у всех есть общий набор функций, который можно реализовать: генерация вопросов письменной формы. Вопросы генерируются с использованием случайных чисел, которые определяют тип возвращаемого результата, количество параметров функции, являются ли параметры входными или выходными параметрами и тип данных параметров. Студент редактирует свой ответ на поставленный вопрос, и его ответ автоматически проверяется с использованием регулярных выражений, как в примере с приложением Java. У каждого типа вопроса будут собственные шаблоны, как с примерами в CQG.

2 Проектирование и реализация приложения

2.1 Структура приложения

На рисунке 4 представлена диаграмма классов. При объектно-ориентированном проектировании были выделены ключевые интерфейсы, что дает возможность в дальнейшем расширять систему без изменения существующего программного кода.

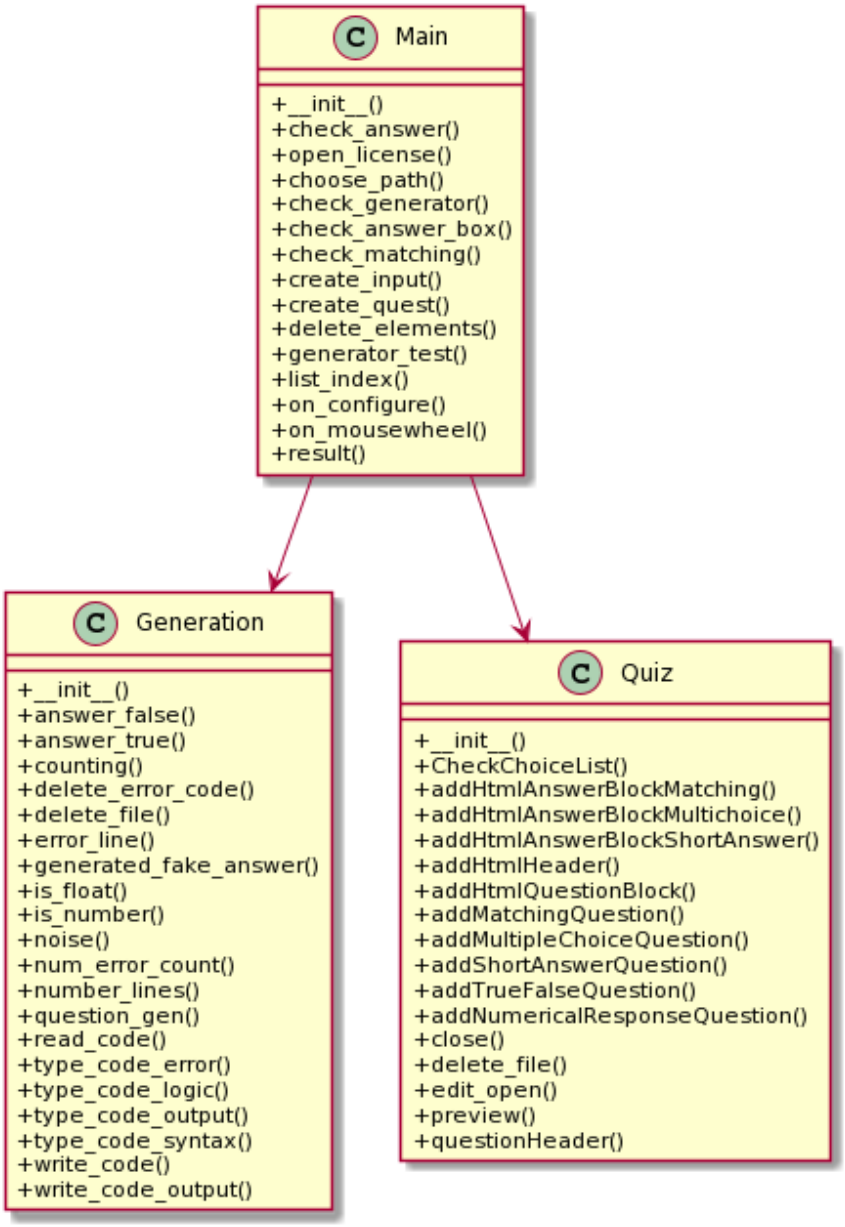

Рисунок 4 – Диаграмма классов

Класс Main содержит основное окно графического интерфейса, основные функции для взаимодействия с генерацией теста и сохранения файлов Moodle XML и HTML.

Метод generator_test() инициализирует основные переменные для взаимодействия с окнами и списками, в которые добавляются данные вопросов, ответов для последующей их проверки.

Метод create_quest() начинает цикл с заданным количеством вопросов пользователем, где вызывается генерация теста в классе Generation. Из этого класса присылаются сгенерированные данные кода C++, вопрос и ответ. Проверяется переданный ответ числовой, текстовый или список и создается случайным образом в зависимости от ответа тип графического ответа для

предварительного просмотра: короткий ответ, множественная выборка или соответствие.

Метод check_matching() проверяет ответы на соответствие.

Метод check_answer_box() проверяет ответы с множественной выборкой.

Метод check_answer() проверяет строковый (короткий) ответ.

Метод result() проверяет, ответил ли пользователь на все вопросы, и возвращается к главному окну генерации теста, предварительно очистив все старые данные окон и переменных.

Класс Generation содержит основные функции генерации вопроса. Тип вопроса, код C++ из директории cods выбираются случайным образом после чего вычисляются данные ответа, вопроса и отформатированного кода C++ и отсылаются в класс Main.

Метод read_code() форматирует синтаксис файла кода C++ и преобразует в конечный вариант кода для компиляции.

Метод type_code_output() форматирует синтаксис для 3-го типа вопроса (при каком входном значении программа выдаст соответствующий результат после выполнения программы).

Метод type_code_syntax() создает одну или несколько синтаксических ошибок в отформатированном коде C++.

Метод type_code_logic() создает одну или несколько логических ошибок.

Метод write_code() записывает отформатированный код в файл quest.cpp и возвращает функцию для выполнения фонового вызова компилятора, чтобы его скомпилировать.

Метод write_code_output() записывает отформатированный код в файл quest.txt для предварительного просмотра, обработка строк кода производится в функции number_lines().

Метод answer_true() компилирует код без ошибок, чтобы выдать результат выполнения кода.

Метод answer_false() компилирует код с ошибками и выдает результат ошибки и строчки кода, на которых ошибки.

Класс Quiz содержит генерацию файла HTML для предварительного просмотра в браузере после решения теста, а также Moodle XML для добавления теста в систему Moodle.

Метод addMatchingQuestion() добавляет вопрос на соответствие с выпадающим списками ответов.

Метод addShortAnswerQuestion() добавляет вопрос с текстовым полем ответа.

Метод addTrueFalseQuestion() добавляет вопрос с типом Да/Нет.

Метод addMultipleChoiceQuestion() добавляет вопрос с множественной выборкой ответов.

Функции addHtmlHeader(), addHtmlQuestionBlock(), addHtmlAnswerBlockMatching(), addHtmlAnswerBlockMultichoice(), addHtmlAnswerBlockShortAnswer() добавляют информацию в файл quiz.html, предварительно форматируя данные тегов HTML, чтобы отделить их от основного кода C++.

Метод questionHeader() добавляет информацию в файл quiz.xml, предварительно форматируя данные тегов XML, чтобы отделить их от основного кода C++.

2.2 Графический интерфейс

Для реализации графической части была использована библиотека tkinter на python. Для разработки интерфейса использовались стандартные элементы управления: tk.Button (кнопки), tk.Label (текстовая метка), Spinbox (двунаправленный счетчик), tk.Entry (поле ввода), tk.OptionMenu (выпадающий список), tk.CheckBox (флажок), tk.Frame (область окна), tk.ScrollBar (полоса прокрутки), tk.Canvas (холст для рисования объектов).

Диаграмма классов графического интерфейса приложения показана на рисунке 5.

На рисунке 6 изображено главное окно программы. В нем создается графическое окно frame_out, в которое помещается двунаправленный счетчик spinbox с указанием количества вопросов и кнопка btn_gen с надписью «Сгенерировать тест».

На рисунке 7 изображено окно предварительного просмотра сгенерированного теста. В нем предусмотрен холст для рисования объектов canvas, в котором находится графическое окно frame_in, содержащее полосу прокрутки страницы по координате Y scrolly. Окно frame_in содержит управляющие функции on_configure(event), которая отвечает за прокрутку страницы с помощью полосы прокрутки, и on_mousewheel(event), которая отвечает за прокрутку страницы колесиком мыши. В frame_in создаются графические окна с отформатированным кодом C++, вопросом, видом ответа на вопрос и кнопкой. Эти окна хранятся в Main в списке в атрибуте frame, а кнопки — в списке в атрибуте button_answer.

После того, как в вопросе нажата кнопка «Отправить», эта кнопка удаляется. После удаления всех кнопок и проверки вопросов всплывает уведомление, в котором содержится количество всех вопросов score_count и

количество вопросов score, на которые пользователь ответил правильно. На рисунке 8 изображено уведомление о проверке результата тестирования.

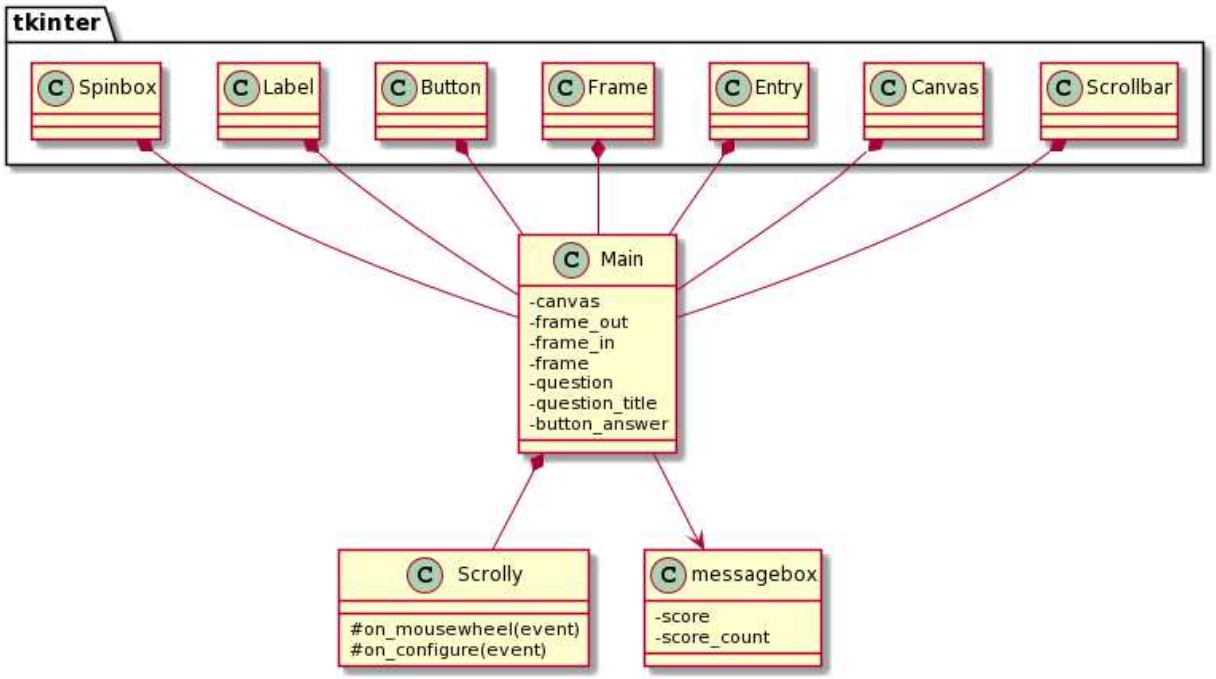

Рисунок 5 – Диаграмма классов графического интерфейса

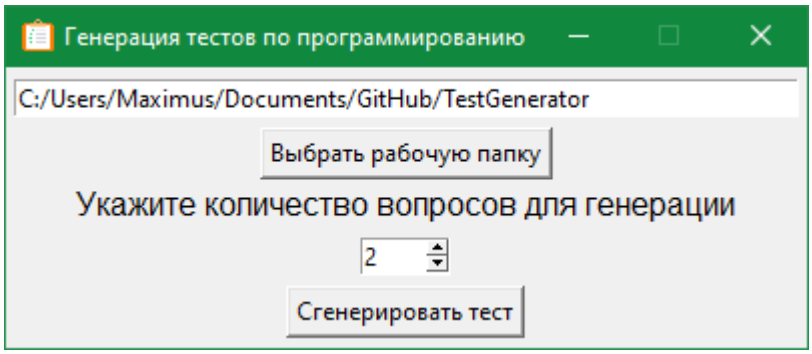

Рисунок 6 – Главное окно программы

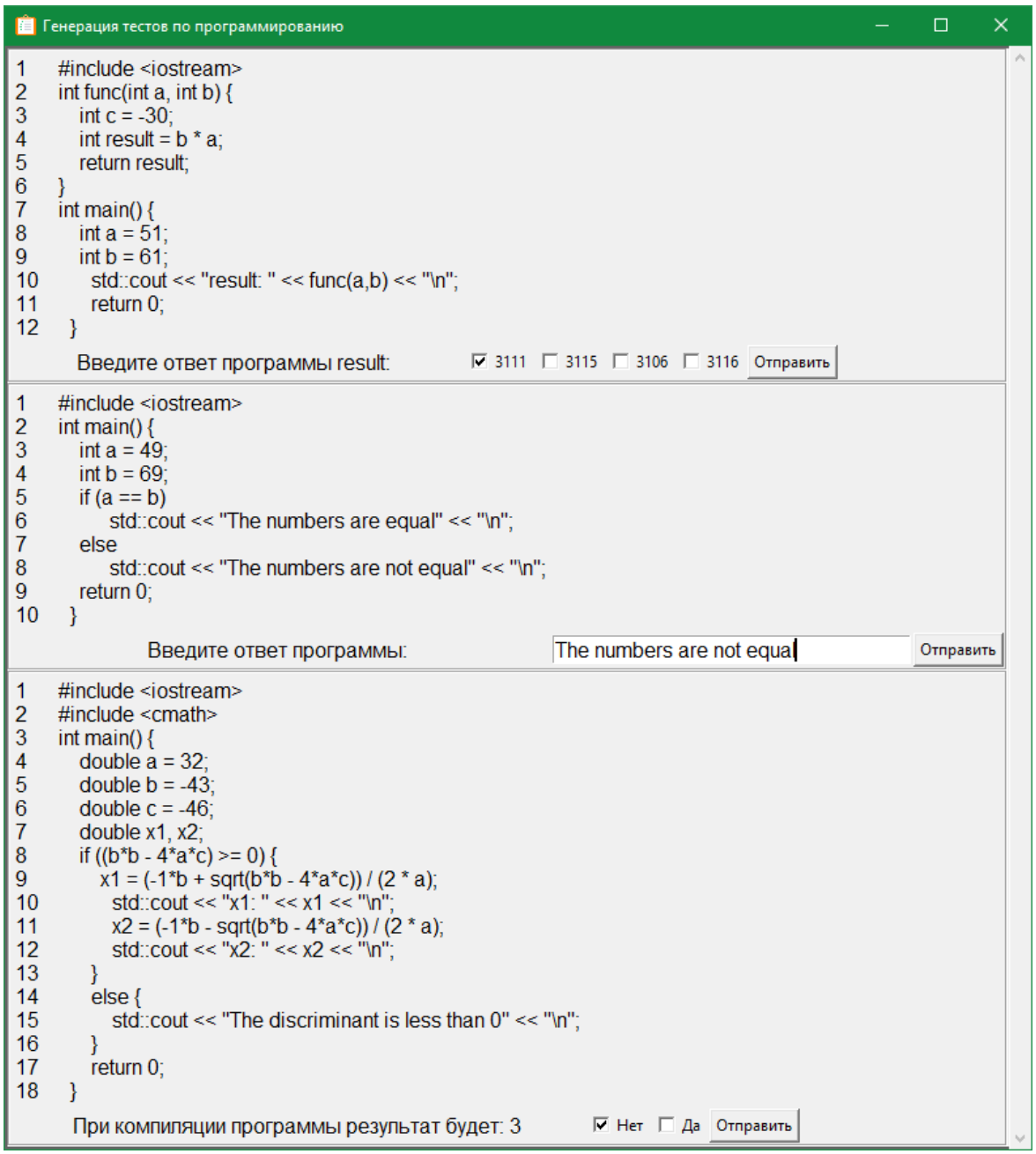

Рисунок 7 – Предварительный просмотр сгенерированного теста

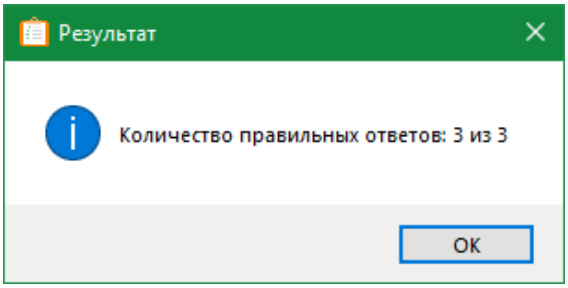

Рисунок 8 – Уведомление о результате тестирования

2.3 Сохранение и загрузка Moodle XML

В приложении для сохранения и загрузки тестовых заданий для системы Moodle используется формат XML. Пример сгенерированного файла Moodle XML приведен в листинге 2. Для работы с форматом XML используется стандартное форматирование текста в Python, а также библиотека Jinja2 для экранирования тегов HTML и XML от основного кода C++. Пример сгенерированного теста в файле HTML приведен на рисунке 9. Пример вопроса из сгенерированного теста в интерфейсе Moodle на рисунке 10.

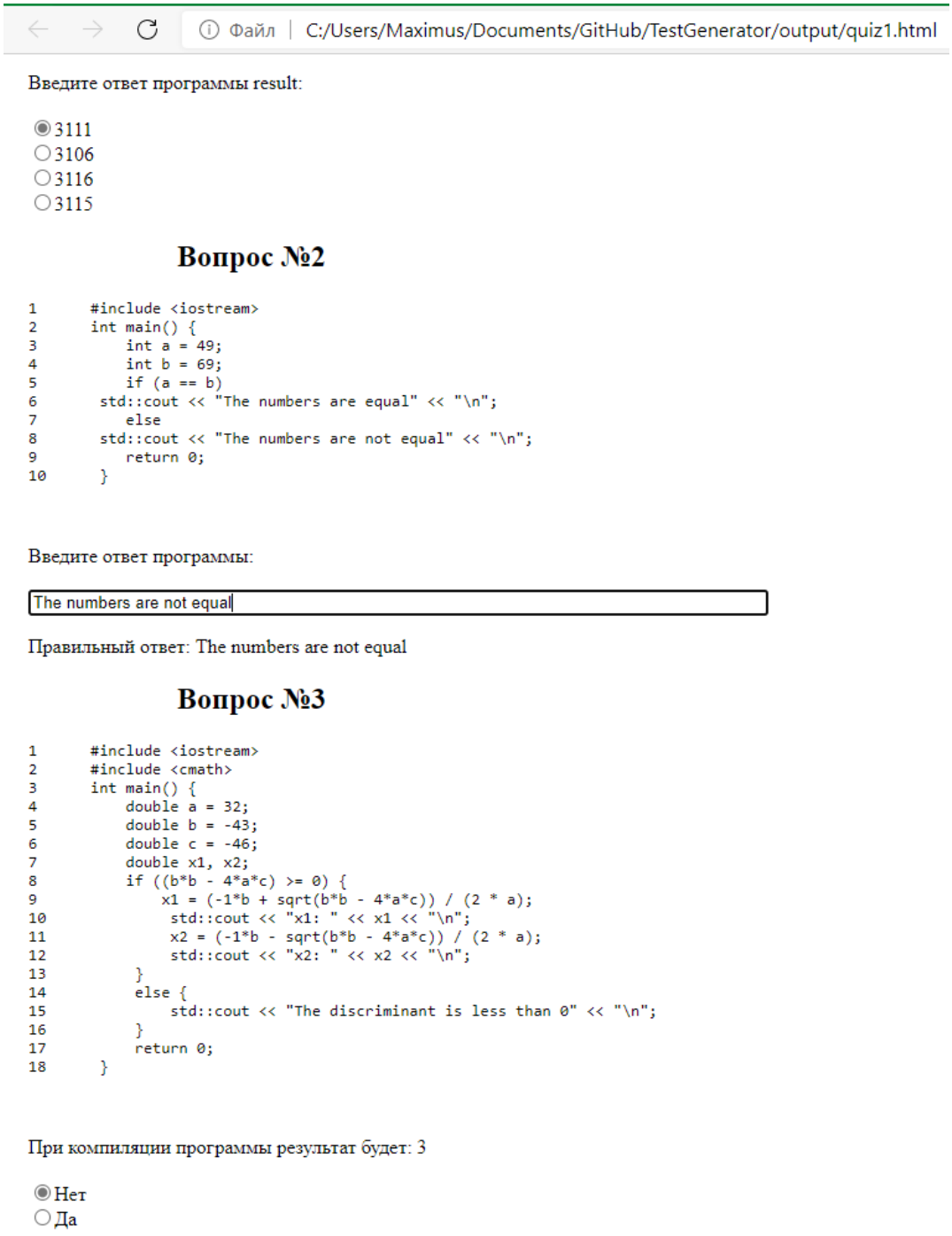

Рисунок 9 – Сгенерированный HTML файл теста

Листинг 2 – Пример сгенерированного Moodle XML с 3 видами вопросов

```
<?xml version="1.0" ?><quiz>
<question type="multichoice">
<name>
 <text>Вопрос №1</text>
</name>
<questiontext format="html">
 <text><![CDATA[<pre>
1    #include &lt;iostream&gt;
2    void func(int a, int b, int c) {
3        int result = 0;
4        while (result &lt; b) {
5            result += a;
6        }
7        std::cout &lt;&lt; &#34;result: &#34; &lt;&lt; result &lt;&lt; &#34;\n&#34;;
8    }
9    int main() {
10        int a = 88;
11        int b = 73;
12        int c = a * b;
13        func(a,b,c);
14        return 0;
15    }
</pre><br /><br />Введите ответ программы result: ]]>
 </text>
</questiontext>
<answer fraction="100"><text>88</text></answer>
<answer fraction="-33.33333" format="html"><text>92</text></answer>
<answer fraction="-33.33333" format="html"><text>89</text></answer>
<answer fraction="-33.33333" format="html"><text>83</text></answer>
<answernumbering>none</answernumbering>
<shuffleanswers>1</shuffleanswers>
<single>true</single>
</question>
<question type="shortanswer">
<name>
 <text>Вопрос №2</text>
</name>
<questiontext format="html">
 <text><![CDATA[<pre>
1    #include &lt;iostream&gt;
2    int main() {
3        int a = 31;
4        int b = -5;
5        if (a == b)
6            std::cout &lt;&lt; &#34;The numbers are equal&#34; &lt;&lt; &#34;\n&#34;;
7        else
8            std::cout &lt;&lt; &#34;The numbers are not equal&#34; &lt;&lt; &#34;\n&#34;;
9        return 0;
10    }
</pre><br /><br />Введите ответ программы: ]]>
 </text>
</questiontext>
<usecase>0</usecase>
<answer fraction="100" format="moodle_auto_format"><text>The numbers are not equal</text></answer>
</question>
<question type="matching">
```

```
<name>
 <text>Вопрос №3</text>
</name>
<questiontext format="html">
 <text><![CDATA[<pre>
1    #include &lt;iostream&gt;
2    #include &lt;cmath&gt;
3    int main() {
4        double a = 35;
5        double b = 27;
6        double c = 0;
7        double x1, x2;
8        if ((b*b - 4*a*c) &gt;= 0) {
9            x1 = (-1*b + sqrt(b*b - 4*a*c)) / (2 * a);
10           std::cout &lt;&lt; &#34;x1: &#34; &lt;&lt; x1 &lt;&lt; &#34;\n&#34;;
11           x2 = (-1*b - sqrt(b*b - 4*a*c)) / (2 * a);
12           std::cout &lt;&lt; &#34;x2: &#34; &lt;&lt; x2 &lt;&lt; &#34;\n&#34;;
13       }
14       else {
15           std::cout &lt;&lt; &#34;The discriminant is less than 0&#34; &lt;&lt; &#34;\n&#34;;
16       }
17       return 0;
18   }
</pre><br /><br />Сопоставьте решения: ]]>
 </text>
</questiontext>
<subquestion><text>x1</text><answer><text>0</text></answer></subquestion>
<subquestion><text>x2</text><answer><text>-0.77</text></answer></subquestion>
<shuffleanswers>true</shuffleanswers>
</question>
</quiz>
```

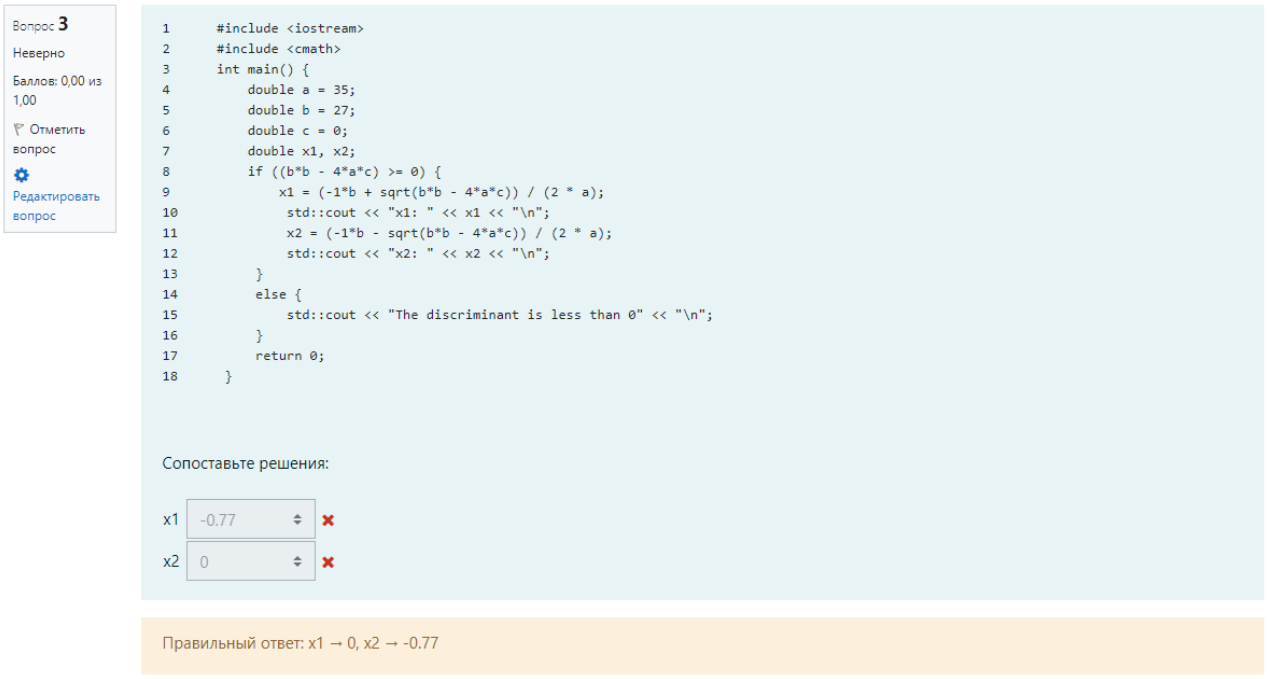

Рисунок 10 – Пример вопроса из сгенерированного теста в интерфейсе Moodle

2.4 Алгоритм генерации тестовых заданий

После перехода в окно предварительного просмотра, для теста генерируется код C++, который считывается из случайного выбранного текстового файла в директории cods, а также генерируется случайно один из 5 типов вопроса. Данные для отображения кода в предварительном просмотре обрабатываются функцией number_lines(), нумерующей строчки кода.

2.4.1 Вопрос со сгенерированной синтаксической ошибкой

Функция type_code_syntax() выбирает случайным образом вид ошибки и если такой ошибки в коде нет, то вызывается стандартная функция type_code_error(), где случайным образом выбирается один из параметров error_syntax в конфигурационном файле, а также выбирается случайным образом метод удаления или изменения этой ошибки. Например, хотим удалить «;». Для начала подсчитывается, сколько всего найденных символов в коде для создания ошибки. Далее выбираем случайно по счету номера найденного символа в коде из всего количества ошибок. Например, ошибок 5, случайно выбираем 1, 3 и 5 номера, следовательно, символ «;» удалится, изменится или удвоится в этих местах кода по счету. Это приведет к ошибкам кода после компиляции. По сравнению с компиляцией кода без ошибок, фоновый процесс в этом случае не создает исполняемый файл для запуска программы, т.к. в коде присутствуют ошибки для компиляции, но результат, что показывает компилятор, оформляется в виде списка ошибок для ответа на вопрос

`«строка с ошибкой : номер ошибки»`

2.4.2 Вопрос со сгенерированной логической ошибкой

Функция type_code_logic() действует по тому же алгоритму поиска и генерации ошибки, что и генерация синтаксических ошибок, только создавая уже логические ошибки. Если создание ошибки не получается, выбирается стандартная функция type_code_error() создания ошибки.

2.4.3 Вопрос, при каком входном значении программа выдаст соответствующий результат после выполнения кода

Функция read_code_output() форматирует синтаксис кода C++ с одним исключением того, что в файле quest.cpp для компиляции записывается весь результат форматирования, а для файла предварительного просмотра кода quest.txt одно из входных значений (число, знак вычисления, слово из словаря)

заменяется на неизвестное значение. Таким образом происходит выполнение программы и берётся результат выполнения. Вопрос выглядит следующим образом: «При каком значении программа выдаст результат». В ответ передается случайное сгенерированное входное значение.

2.4.4 Вычисление результата выполнения показанного кода

Функция read_code() форматирует код C++ с заданными параметрами синтаксиса (генерация чисел, знаков, слов и шумов), после чего этот код сохраняется в файл quest.cpp и вызывается subprocess (фоновый процесс), благодаря чему вызов компилятора происходит в фоновом режиме. Для данного типа вопроса уже заранее известно, что код не содержит ошибок после компиляции, в ином случае этот код не пройдет проверки и генерация данного вопроса будет отменена. После компиляции кода будет взят результат программы и отформатирован в функции answer_true(), где ответ может быть как одним значением, так и списком ответов с присвоенным значением.

2.4.5 Вопрос Да/Нет

Работает по принципу 4-го типа вопроса, но берет только правильный ответ. Если передаётся список ответов, то из этого списка выбирается случайным образом один ответ. Дальше из двух вариантов случайно выбирается, какой из ответов станет верным «Да» или «Нет», т.е. в случае «да» задается вопрос с верным вариантом ответа скомпилированной программы, а в случае «нет» вызывается функция с генерацией неправильного ответа. В классе Main создается множественная выборка из вариантов «да/нет».

2.4.6 Описание классов

Класс Main принимает сгенерированный вопрос и ответ, создает вопрос с одним из видов ответа (короткий ответ, множественная выборка, соответствие) на основе сгенерированного вопроса из класса Generation. Отправленный ответ или ответы сравниваются с введенными или выбранными данными пользователем в предварительном просмотре теста и проверяется ответ на правильность, после чего считается результат правильных ответов.

Короткий ответ — это строковое поле с результатом программы, номером строки ошибки, либо входным значением для 3-го типа вопроса.

Множественная выборка создается по принципу, что нам известен правильный результат. Если ответ один, то генерируются оставшиеся неправильные ответы для выборки из параметра count_multichoice в конфигурационном файле. Например, задано 4 варианта выборки, значит один

ответ правильный, а в оставшихся трех вариантах генерируются ответы максимально приближенные к правильному. Если это число, то формируется список в диапазоне от минимального до максимального значения, приближенные к этому числу. Диапазон прописывается в конфигурационном файле. После этого список перемешивается и добавляются неправильные ответы в оставшиеся 3 варианта. Если присылается список ответов, то правильные ответы заносятся первыми в очередь, затем их количество умножается в два раза, чтобы из оставшейся выборки сгенерировать неправильные ответы по тому же алгоритму с множественной выборкой, только теперь эти неправильные ответы генерируются исходя из каждого правильного ответа в списке. Для предварительного просмотра, когда создается выборка, их индексы перемешиваются, чтобы эти ответы разместить случайным образом.

Соответствие значений и ответов составляется только в том случае, если ответ пришел списком, т.е. в нем есть формат «значение: ответ». Для каждого такого элемента из списка пишутся значения и возле них строится выпадающее меню со всеми вариантами ответа, предварительно перемешанными.

Класс Quiz регистрирует данные вопросов и ответов, затем сохраняет их в файлы HTML и XML в директории output.

3 Инструкции

3.1 Инструкция пользователя

После запуска основного файла main.py открывается главное окно, где нужно выбрать путь к рабочему каталогу с конфигурационным файлом generator.conf. В данном файле прописаны все настройки и конфигурации для программы. В параметрах значения нужно прописывать строго по инструкции. Если предусматривается одно значение, то запись выглядит следующим образом:

```
'параметр': значение
```

Если предусматривается диапазон значений, то запись выглядит следующим образом:

```
'параметр': [значение_1, значение_2]
```

Если значение имеет словесный вид, то запись выглядит следующим образом (это относиться и к диапазону значений):

```
'параметр': 'значение'
```

Каждый параметр задается через запятую, кроме завершающего параметра. В готовом конфигурационном файле из репозитория даны комментарии, какой параметр за что отвечает.

Если конфигурационный файл найден и в нем прописаны названия директорий для работы с кодами на C++ (параметр path_cods), входных и выходных файлов для запуска фрагментов кода (параметр files) и выходных файлов HTML и XML (параметр path_output), то в главном окне появляется выбор количества вопросов для генерации тестов. Параметр, отвечающий за диапазон количества вопросов: count_question. Нажимаем кнопку «Сгенерировать тест» и переходим в окно предварительного просмотра теста.

На этом этапе в директории output сгенерировались два файла: quiz_<номер_теста>.html и quiz_<номер_теста>.xml. В файле HTML содержится сгенерированный тест, но в нем уже указаны ответы на каждый вопрос для предварительной проверки результатов. В файле quiz.xml содержится сгенерированный файл для тестирования в формате Moodle XML.

Чтобы загрузить файл в формате Moodle XML в систему Moodle, необходимо войти в систему под учетной записью преподавателя или администратора, перейти в банк вопросов и импортировать сгенерированный файл. После этого можно создать тест и добавить вопросы из банка вопросов.

В предварительном просмотре теста можно его выполнить. Ответы бывают 5 типов: короткий ответ, множественный выбор с одним верным ответом из указанного в конфигурации количества (параметр: count_multichoice), множественный выбор с несколькими верными вариантами, вопрос «Да/Нет», вопрос на соответствие.

Сами вопросы бывают 5 типов: вопрос со сгенерированной синтаксической ошибкой, вопрос со сгенерированной логической ошибкой, вопрос при каком входном значении программа выдаст соответствующий результат после компиляции, вычисление результата выполнения показанного кода, проверка ответа на истину.

После ввода ответа в поле нажимаем кнопку «Отправить» и происходит проверка ответа на вопрос. Если ответ оказался правильный, то поле вопроса окрашивается в зеленый цвет, если нет – в красный. Кнопка «Отправить» после ответа исчезает.

После того, как ответили на все вопросы, всплывает уведомление о проверке результатов тестирования. После закрытия уведомления, окно предварительного просмотра теста закрывается и появляется главное окно генерации теста. После этого можно повторять процедуру генерации нового теста.

Шаблоны фрагментов кода для генерации тестов находятся в директории cods в текстовом формате. Код может содержать специальные директивы, влияющие на формирование итогового случайного фрагмента кода:

а) генерация случайных чисел: {number}. Для изменения генерируемого числа меняем диапазон значениий параметра в конфигурационном файле: generated_number;

б) генерация знака числового оператора: {action}. Для изменения знаков операторов меняем диапазон данных параметров в конфигурационном файле: sign_for_action;

в) генерация слова из словаря: {dictionary}. Для изменения или добавления строк меняем данные параметров в конфигурационном файле: dictionary;

г) генерация одной буквы из сгенерированного слова из словаря: {letter}.

Пример:

Входной файл, что напишет составитель кода: int a = {number};

Сгенерированный результат программой: int a = 69;

Также в коде можно создать альтернативу – случайный выбранный фрагмент кода из предложенного закрытого диапазона. Синтаксис для создания шумов (случайных изменений кода) выглядит следующим образом:

{-}Строчка_кода – начало альтернативы.

{^}Строчка_кода – вариант выбора альтернативы.

{*}Строчка_кода – конец альтернативы.

Пример кода:

```
{-}while (result < a) {
    result += b;
}
{^}while (result < b) {
    result += a;
 }
 {^}while (result != c) {
    result++;
 }
 {*}return result;
```

Преобразуется в сгенерированный код:

```
while (result != c) {
    result++;
}
return result;
```

Подробные настройки параметров для конфигурационного файла описаны ниже.

Параметр type_quiz отвечает за номер генерируемого типа вопроса:

а) 0 – по умолчанию случайный тип вопроса;

б) 1 – генерация логической ошибки;

в) 2 – генерация синтаксической ошибки;

г) 3 – генерация входного значения при известном результате выполнения программы;

д) 4 – выходной результат программы после выполнения;

е) 5 – тип вопроса «Да/Нет».

Параметр code отвечает, какой файл кода на C++ выбирается в директории cods. По умолчанию 0 – случайный файл кода на C++ в директории cods.

Параметр error задает настройки генерации синтаксических и логических ошибок. После форматирования синтаксиса шаблона кода выбирается случайным образом один из алгоритмов генерации одна или несколько ошибок в коде. Стандартный алгоритм генерации выбирает случайным образом значения из параметра error, ищет схожие символы с этим значением в коде, затем удаляет или заменяет на другой символ одну или несколько найденных символов случайным образом. Для остальных алгоритмов, не использующие значения из параметра error, принцип генерации ошибок схож по смыслу замены или удаления фрагмента кода. Также для некоторых алгоритмов генерация ошибок задаётся вручную, если заведомо известно, что при выполнении программы компилятор не определит фрагмент кода как ошибку.

Если ответ дробный числовой, то в конфигурационном файле прописано количество знаков после запятой при округлении: fractional_number.

Пример: fractional_number: 2. Число 2.1234 преобразуется в число 2.12.

Параметр number_answer отвечает за генерацию неправильных ответов в множественной выборке на основе известного правильного ответа в заданном диапазоне.

Пример: number_answer: [5,6].

Если ответ равен 12, то программа сгенерирует числа в диапазоне от 7 (12-5) до 18 (12+6) и выберет значения из этого диапазона. По этой же логике параметр fractional_number_answer работает для дробных чисел.

Параметр type_quiz_exception относится к 3 типу вопроса (при каком входном значении программа выдаст соответствующий результат после компиляции). Так как вопрос довольно специфичный и при составлении теста может случайно выбраться код, не подходящий для вопроса, из-за чего точный

ответ дать невозможно, то необходимо прописать названия файлов без расширения, которые подходят для данного типа вопроса в этот параметр.

Параметр number_of_generated_numbers отвечает за количество сгенерированных чисел при генерации случайного входного файла. Например, в вопросе нужно посчитать количество положительных чисел, записанных в текстовом файле in.txt. Тогда в файле генерируется заданное этим параметром количество чисел.

Важный момент: при компиляции программы все ответы при выводе (cout, printf и т.д.) должны иметь формат: «cout << "значение" : "ответ" << "\n"». В конце вывода нужно выводить символ новой строки ("\n").

3.2 Инструкция разработчика

Для работы с исходным кодом и его компиляцией необходимо иметь компоненты:

- Python 3.9.x или последняя версия;
- компилятор g++ (GCC).

Для системы Windows необходимо прописать PATH для компилятора g++.

Если какие-то библиотеки будут отсутствовать при компиляции исходного кода, то установите их согласно методике инструкции в зависимости от вашей системы.

Скачиваем репозиторий проекта с github [7].

Переходим в директорию проекта: cd TestGenerator.

Для запуска программы необходимо запустить файл main.py с помощью интерпретатора Python.

Шаблоны фрагментов кода для генерации тестов хранятся в директории cods. В директории files находятся входные и выходные файлы для запуска фрагментов кода. В директории output сохраняются сгенерированные тесты в форматах HTML и XML.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы было разработано приложение для персональных компьютеров — генератор тестовых заданий по программированию на языке C++.

В разработанном приложении были реализованы следующие функции:

а) создание теста со случайным типом вопроса;

б) созданы 5 типов вопросов: вопрос со сгенерированной синтаксической ошибкой, со сгенерированной логической ошибкой, при каком входном значении программа выдаст указанный результат, какой результат выдаст показанный код, верно ли указан результат выполнения;

в) предварительный просмотр сгенерированного теста в программе, где возможно сразу ответить на вопросы и узнать результат тестирования;

г) для каждого типа вопроса были созданы 5 видов ответа на вопрос: короткий ответ, множественный выбор с одним верным ответом, множественный выбор с несколькими вариантами ответа, «Да/Нет», на соответствие;

д) генерация файла HTML для просмотра теста в браузере и генерация Moodle XML для импорта в LMS Moodle.

Предусмотрена гибкая настройка генерации тестов.

Приложение можно улучшить следующим образом:

а) добавить возможность генерации нескольких ошибок в коде вопроса для некоторых алгоритмов генерации ошибок;

б) добавить возможность исправления ошибки в коде вопроса.

Последняя версия приложения размещена в git-репозитории [7].

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Moodle - Open-Source learning platform. – Режим доступа: https://moodle.org (дата обращения: 05.06.2021).
2. Hoffman D., Giannelia F., Lu M. Computational Quiz Generation // Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS). – The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2014. – С. 1. – Режим доступа: http://worldcomp-proceedings.com/proc/p2014/FEC2290.pdf (дата обращения: 24.12.2020).
3. Ene A., Știrbu C. Automatic generation of quizzes for C programming language // 2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI). – IEEE, 2020. – С. 1-4. – Режим доступа: https://ieeexplore.ieee.org/abstract/document/9223159 (дата обращения: 24.12.2020).
4. Swing APIs and Developer Guides. – Режим доступа: https://docs.oracle.com/javase/8/docs/technotes/guides/swing/index.html (дата обращения: 05.06.2021).
5. Qt | Cross-platform software development for embedded and desktop. – Режим доступа: https://www.qt.io/ (дата обращения: 05.06.2021).
6. tkinter - Python interface to Tcl/Tk - Python 3.9.5 documentation. – Режим доступа: https://docs.python.org/3/library/tkinter.html (дата обращения: 05.06.2021).
7. GitHub - MrByMaximus/TestGenerator. – Режим доступа: https://github.com/MrByMaximus/TestGenerator (дата обращения: 17.04.2021).

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

Кафедра вычислительной техника

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись
«07» 06 2021 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 – «Информатика и вычислительная техника»

Генератор тестовых заданий по программированию

Руководитель _____ 07.06.2021 ст. преподаватель К. В. Пушкарев
подпись, дата

Выпускник _____ 02.06.2021 М. В. Гильденберг
подпись, дата

Нормоконтролер _____ 07.06.2021 К. В. Пушкарев
подпись, дата

Красноярск 2021