УДК 378.147.88

АВТОМАТИЗИРОВАННАЯ ПРОВЕРКА ЛАБОРАТОРНЫХ РАБОТ ПО ПРОГРАММИРОВАНИЮ

А.В. Ильин

Белорусский государственный университет
пр. Независимости, 4, 220030, г. Минск, Беларусь, ilyin@lyceum.by

На факультете прикладной математики и информатики Белорусского государственного университета (ФПМИ БГУ) в течение последних лет была проведена работа по внедрению современных подходов в процесс проверки и оценки лабораторных работ по дисциплине «Программирование» (1–2 семестры). В данной статье собраны ключевые моменты, описывающие опробованные системы (iRunner, Яндекс.Контест, AnyTask), их возможности, преимущества и недостатки. Кроме того, приведена информация о том, как можно интегрировать популярный в индустрии механизм юнит-тестирования с упомянутыми системами автоматизированной проверки. Данная статья будет полезна всем преподавателям дисциплины «Программирование» (и смежных с ней дисциплин), которые хотят автоматизировать процесс проверки лабораторных работ и/или начать использовать онлайн-системы в образовательном процессе.

***Ключевые слова*:** Программирование; методы обучения; автоматическое тестирование; юнит-тестирование; iRunner, Яндекс.Контест; AnyTask.

AUTOMATED TESTING OF PROGRAMMING LABORATORY ASSIGNMENTS

A. Ilyin

Belarusian State University
4 Niezalieznasci Avenue, Minsk 220030, Belarus, ilyin@lyceum.by

During the past few years at the Faculty of Applied Mathematics and Informatics of the Belarusian State University (FPMI BSU), work has been carried out to introduce modern approaches to the process of checking and evaluating laboratory assignments in the discipline “Programming” (1–2 semesters). This article contains key points describing the tested systems (iRunner, Yandex.Contest, AnyTask), their capabilities, advantages and disadvantage. It also provides information on how you can integrate the industry's popular unit testing engine with the automated test systems mentioned above. This article will be useful to all teachers of the discipline “Programming” (and related disciplines) who want to automate the process of checking laboratory work and/or start using online systems in the educational process.

***Key words*:** Programming; teaching methods; automated testing; unit testing; iRunner; Yandex.Contest; AnyTask.

Введение. В наши дни актуальным вопросом является применение современных технологий и подходов в различных видах деятельности, в том числе и в образовательном процессе. Сегодня в технологических компаниях повсеместно применяются механизмы автоматизированного тестирования разрабатываемого программного обеспечения. Такие подходы позволяют, с одной стороны, повысить качество проверки кода, а, с другой, освобождает сотрудников от необходимости выполнять монотонную работу.

На факультете прикладной математики и информатики Белорусского государственного университета (ФПМИ БГУ) в течение последних лет была проведена работа по внедрению упомянутых механизмов в учебный процесс на примере дисциплины «Программирование» (1–2 семестры), в которой в этих семестрах изучается язык программирования C++ и язык ассемблера. За это время были опробованы несколько различных систем – iRunner (acm.bsu.by), Яндекс.Контест (contest.yandex.ru), AnyTask (anytask.org), EduFPMI (edufpmi.bsu.by). Как показала практика, ни одна из рассмотренных систем не смогла полностью покрыть необходимый функционал. Кроме того, для интеграции библиотеки юнит-тестирования и тестирующих систем потребовались дополнительные усилия.

В общем виде процесс сдачи лабораторной работы начинается с отправки студентами решений в систему автоматизированной проверки. Результаты тестирования формируют часть от общей суммы баллов за лабораторную работу. После наступления крайнего срока сдачи, когда дальнейшая отправка решений запрещена, последние отправленные версии каждого из участников проходят ручной просмотр – в этот момент проверяется то, что невозможно формализовать и автоматизировать (оптимальность идеи, качество исходного кода и т. п.). По итогу проверки студентам добавляется оставшаяся часть баллов. Помимо того, по всем или части лабораторных работ практикуются беседы преподавателя со студентом, в ходе которого можно проверить понимание студентом каких-то моментов и самостоятельность выполнения работы.

Общие сведения о механизме тестирования. Для автоматизированной проверки заданий по программированию был выбран подход под названием «юнит-тестирование» – способ тестирования, широко применяемый в мире промышленного программирования. Он позволяет проверить на правильность отдельные модули исходного кода (функции, классы), а не полноценную программу, как это принято, например, в спортивном

программировании. В качестве библиотеки юнит-тестирования, в силу различных причин, была выбрана популярная библиотека Google Test [1] с открытым исходным кодом. Поскольку тестирующие системы (как Яндекс.Контест, так и iRunner) изначально создавались для проведения олимпиад и тренировок к ним, и, как следствие, не имеют встроенной поддержки юнит-тестов, для интеграции была создана специальная прослойка, код которой расположен в открытом репозитории [2] вместе с инструкцией по использованию, а также примером лабораторной работы, использующей данную прослойку.

Скриптовая прослойка дополняет функционал автоматизированных систем проверки, определяя алгоритм тестирования и оценки. Настройка системы оценивания производится путём редактирования специального файла *tester_config.py*, в котором указываются имена тестов и баллы за них. Файл позволяет также настраивать видимость названий отдельных тестов: по умолчанию, студентам видны лишь названия групп, но не конкретных тестов в них (например, «QuickSort.*»), но можно включить отображение конкретного теста в отдельности от группы («QuickSort.Sample»), что актуально, в основном, для заранее предоставленных участнику примеров.

Отдельный файл *build.sh* определяет алгоритм компиляции решений. С его помощью, в числе прочего, можно запускать средства анализа исходного кода, организовывать компиляцию многофайловых проектов и фильтровать лог ошибок компилятора. В наших лабораторных в рамках анализа исходного кода проверяется его соответствие требуемому стилю оформления. Помимо этого, возможно также подключение средств для нахождения других недочётов (неоптимальных конструкций, неиспользуемого кода и т. п.), однако на практике данная возможность пока что остаётся невостребованной. Последняя функция (фильтрация ошибок) используется для того, чтобы участник не мог посмотреть имена имеющихся на сервере тестов через ошибки компиляции и таким образом понять, что же именно будет проверяться.

Отдельно стоит отметить использование при проверке санитайзеров – специальных режимов компиляции, при которых во время запуска программы можно обнаружить часть ошибок, даже если они не повлияли на итоговый результат выполнения. В режиме по умолчанию, каждый тест запускается по два раза в четырёх редакциях – debug-версия (O0), оптимизированная версия (O2), режима проверки обращений к адресному пространству (Address Sanitizer) и проверки управления памятью (Memory Sanitizer). Тест считается пройденным тогда и только тогда, когда все 8 запусков были успешными.

Отметим, что, помимо различных средств интеграции с системами автоматического тестирования, с помощью созданных скриптов имеется возможность проведения автоматизированной оффлайн-проверки, например, на компьютере преподавателя. В результате запуска всё тех же тестов, формируются индивидуальные отчёты о тестировании и сводная таблица с баллами в формате CSV. Такой подход используется, например, в части курса, посвящённой языку ассемблера, когда появляется необходимость проверки работ студентов, написанных для разных операционных систем – исходный код для ассемблерный не является кросс-платформенным, а тестирующие системы, как правило, могут запускать решения лишь под одной конкретной ОС.

После завершения подготовки тестов, скрипта компиляции и выставления баллов за тесты, с помощью отдельного специального скрипта подготавливается архив для тестирующей системы. Следует заметить, что прослойка написана универсальным образом, так что используемый в дальнейшем механизм тестирования (iRunner, Яндекс.Контест, локальный запуск) не влияет ни на сами тесты, ни на сопутствующие конфигурационные файлы, о которых шла речь выше. Как следствие, прослойку можно единообразно использовать для подготовки задач вне зависимости от используемой в дальнейшем системы, а подготовленные тесты, при необходимости, легко переносить из одной системы в другую.

Система тестирования iRunner 2. Система iRunner 2 является разработкой сотрудников ФПМИ БГУ. Она широко применяется для проведения различных олимпиад по программированию, а также активно используется при преподавании дисциплины «Теория алгоритмов» [3], где формат задач схож с олимпиадными. Помимо непосредственно связанного с тестированием программ функционала, система имеет механизмы ведения журнала успеваемости и назначения задач студентам (как общих для всей группы, так и индивидуальных заданий), механизм контроля за временем сдачи (после некоторого крайнего срока получаемые студентом баллы автоматически снижаются в 2 раза), разделение на подгруппы и электронную очередь, а также систему для проведения опросов студентов по теории. Для регистрации учащихся в системе используется информация из внутренней сети БГУ, однако она имеет собственную базу пользователей и механизмы защиты наподобие обязательной смены пароля при первом входе и двухфакторной аутентификации.

Изначально внутренние механизмы iRunner не предусматривали возможность интеграции упомянутой прослойки, однако летом 2020 года

они были немного доработаны и в настоящее время проходят апробацию на лабораторных работах с юнит-тестированием. Отметим, что речь тут идёт исключительно про внутренние механизмы тестирования – внешний вид отчёта для участника аналогичен стандартному для системы и позволяет просматривать результат выполнения тестов (групп тестов), их наименования и полученные баллы (рис. 1).

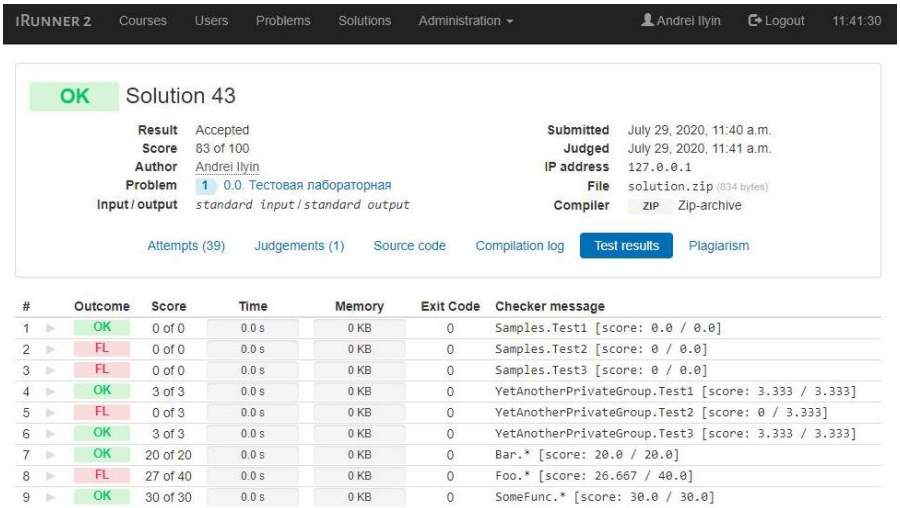


Рис. 1. Пример отчёта о проверке решения лабораторной в iRunner 2

Среди явных преимуществ данной системы стоит отметить удобство разработки и внедрения изменений, поскольку данная система имеет открытый исходный код и запущена на серверах БГУ, к которым имеется необходимый доступ. В частности, мы можем не только оперативно совершенствовать саму систему, но и конфигурировать сторонние элементы – операционную систему и компилятор, библиотеки и программ статического анализа. В совокупности с интегрированными механизмами, специфичными для образовательного процесса, данная система представляется наиболее перспективной для использования в будущем. iRunner – это единая система, которая обладает хорошим функционалом, надёжная и хорошо зарекомендовавшая себя на других дисциплинах, а также при проверке лабораторных по программированию в олимпиадном стиле.

Тем не менее, в системе имеется существенный недостаток – отсутствие полноценной подсистемы для рецензирования кода (code review). Code review – общепринятая в индустрии практика, когда, помимо автоматической проверки на тестах, весь код проходит процесс

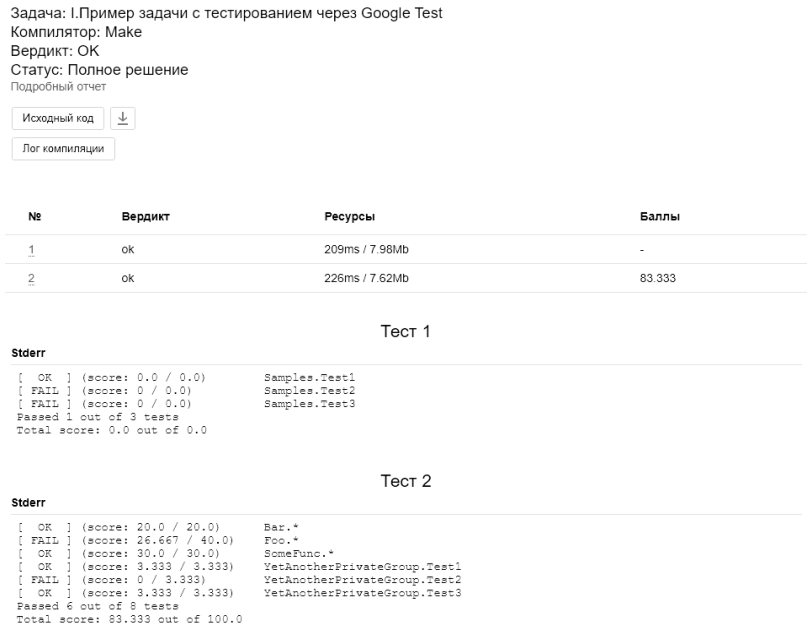
прочтения другими разработчиками, ведутся дискуссии о конкретных его частях и логики программы в целом. На данный момент система позволяет удобным образом читать решения студентов, имеется красивая подсветка синтаксиса и нумерация строчек. Тем не менее, у преподавателя отсутствует возможность написать замечания или комментарии, относящиеся к конкретным строчкам решения, вести дискуссии по конкретным блокам исходного кода. Кроме того, даже существующий механизм отображения решения не поддерживает многофайловые проекты, коими являются решения больше части лабораторных работ по C++. Таким образом, лучшим способом разбора ошибок на сегодняшний день является скачивание решения, его просмотр на локальной системе и объяснение замечаний в ходе последующей живой беседы со студентом. Хотя на первый взгляд это и не кажется плохим алгоритмом, тут имеется ряд недостатков: во-первых, большая часть времени во время беседы тратится не на обсуждение спорных или просто значимых моментов, а на простое зачитывание списка недочётов (а суммарное время бесед ограничено длительностью пары), и, во-вторых, для студента значительно удобнее и полезнее, когда имеется возможность в любой момент пересмотреть список своих ошибок и нет необходимости запомнить все их на слух с первого раза.

Ещё одним узким местом является таблица успеваемости по курсу. На данный момент система рассчитывает, что для каждой из задач оценка бинарная – либо все тесты пройдены, либо баллов за неё не выставляется. Ожидается, что этот недочёт будет в ближайшем будущем исправлен, так что баллы за лабораторные работы будут отображаться и учитываться не только в отчётах о тестировании, но и в сводной таблице.

Системы Яндекс.Контест, AnyTask, ReviewBoard. Система Яндекс.Контест является частичным аналогом системы iRunner и во многих отношениях схожа с первой рассмотренной системой. Она также позволяет проводить автоматическую проверку решений задач, причём уже давно обладает функционалом для тестирования задач по произвольному сценарию. Тем не менее, из-за более простого интерфейса, предоставляемого для интеграции, имеются небольшие нюансы, например, с отображением текстового отчёта о тестировании, который видит студент (напр., имеется ограничение на максимальный отображаемый размер вывода).

Непосредственно система Яндекс.Контест не имеет специфичных для образовательного процесса механизмов, однако их предоставляет другая система – AnyTask, которая содержит в себе функционал по ведению журнала успеваемости и учёту крайних сроков сдачи. Благодаря усилиям

разработчиков, система тесно интегрирована с Яндекс.Контест, так что, когда студент отправляет в AnyTask своё решение, оно может быть автоматически передано на автоматизированное тестирование, результат которого (как отчёт, так и суммарный балл), будут переданы назад. Данная связка хорошо себя зарекомендовала в прошлом, но у неё тоже есть существенный недостаток – если решение лабораторной состоит из нескольких файлов, заархивированных в единый архив, то такие решения AnyTask не может передать на тестирование, даже несмотря на то, что сам Яндекс.Контест нормально их обрабатывает. Этот недочёт в интеграционном коде сильно усложняет жизнь, поскольку требует дополнительных ручных усилий по переброске и синхронизации результатов. К сожалению, в силу различных причин, самостоятельно найти и устранить проблему на данный момент не представляется возможным (рис. 2).


Задача: I.Пример задачи с тестированием через Google Test
Компилятор: Make
Вердикт: OK
Статус: Полное решение
Подробный отчет
Исходный код
Лог компиляции

№	Вердикт	Ресурсы	Баллы
1	ok	209ms / 7.98Mb	-
2	ok	226ms / 7.62Mb	83.333

Тест 1

Stderr

```
[   OK  ] (score: 0.0 / 0.0)        Samples.Test1
[ FAIL ] (score: 0 / 0.0)           Samples.Test2
[ FAIL ] (score: 0 / 0.0)           Samples.Test3
Passed 1 out of 3 tests
Total score: 0.0 out of 0.0
```

Тест 2

Stderr

```
[   OK  ] (score: 20.0 / 20.0)      Bar.*
[ FAIL ] (score: 26.667 / 40.0)     Foo.*
[   OK  ] (score: 30.0 / 30.0)      SomeFunc.*
[   OK  ] (score: 3.333 / 3.333)    YetAnotherPrivateGroup.Test1
[ FAIL ] (score: 0 / 3.333)         YetAnotherPrivateGroup.Test2
[   OK  ] (score: 3.333 / 3.333)    YetAnotherPrivateGroup.Test3
Passed 6 out of 8 tests
Total score: 83.333 out of 100.0
```

Рис. 2. Пример отчёта о проверке решения лабораторной

Последней рассматриваемой в данном блоке системой, с которой также имеется тесная интеграция у AnyTask, является ReviewBoard. Данная система позволяет проводить полноценные дискуссии о блоках исходного кода, писать комментарии к конкретным строчкам – одним словом, проводить то самое полноценное «code review», отсутствие которого мы отмечали, как недостаток iRunner 2 на сегодняшний день. Стоит также отметить, что тут уже отсутствуют проблемы с интеграцией – решения в виде архивов успешно передаются в ReviewBoard, где

автоматически распаковываются и красиво отображаются в веб-интерфейсе (рис. 3).

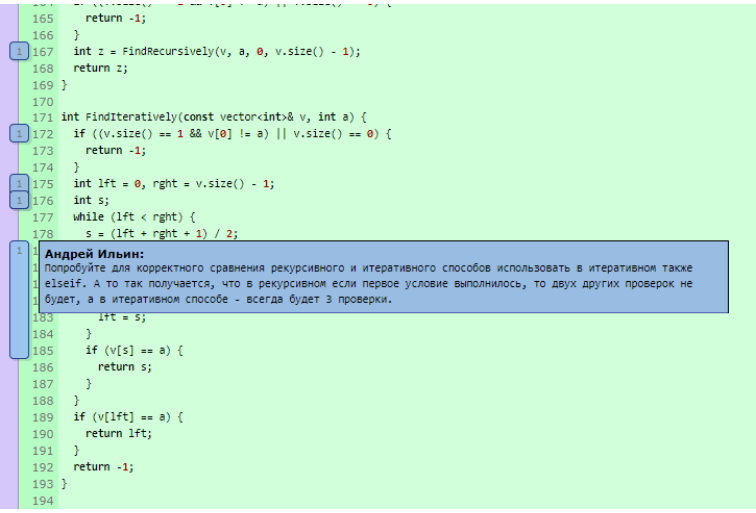


Рис. 3. Пример отображения комментариев в Review Board

Несмотря на достаточно широкий спектр предоставляемого функционала, в данной связке отсутствуют интегрированные функции опроса теоретических знаний (вместо этого предлагается использовать любую внешнюю систему и вручную перенести из неё результаты). Кроме того, системы AnyTask и ReviewBoard, в отличие от iRunner, не поддерживают механизм двухфакторной аутентификации.

Система Moodle (EduFPMI). Данная система не имеет специфичных для преподавания программирования средств наподобие автоматизированной проверки решений или рецензирования исходного кода. Тем не менее, в ней присутствуют общие для всех предметов механизмы: отслеживание успеваемости, организация опросов по теоретическому материалу, возможность сдачи заданий и отслеживание крайнего срока их сдачи. Кроме того, система Moodle позволяет хорошо структурировать содержание курса, группируя теоретические материалы с соответствующими заданиями и отображая это всё в виде единого списка. Таким образом, хотя система и не рассчитана на дисциплину «программирование», она также может применяться в учебном процессе – она хорошо подходит, например, для случаев, когда лабораторные работы тестируются оффлайн, поскольку основная часть отсутствующего в Moodle функционала при таком сценарии всё равно не используется.

Заключение. В ходе проведённого исследования был описан общий подход к организации проверки лабораторных работ по дисциплине «программирование» и смежных с ней по формату. Кроме того, был рассмотрен ряд существующих сервисов, для которых было дано общее описание функционала, выявлены их достоинства и недостатки. Помимо прочего, в данной статье был предоставлен специальный набор скриптов, позволяющих простым образом использовать подход юнит-тестирования вне зависимости от тестирующей системы.

Ожидается, что данные материалы будут полезны всем преподавателям, которые хотят внедрить современные методы автоматизированного тестирования в образовательный процесс, а также разработчикам, которые могут усовершенствовать перечисленные выше узкие места.

Библиографические ссылки

1. Google Test [Электронный ресурс]. Google Inc., 2008–2020. URL: https://github.com/google/googletest. (дата обращения: 18.09.2020.)

2. Ильин А.В., Неверо А.А. Шаблон тестов к лабораторным работам [Электронный ресурс]. 2020. URL: https://github.com/andrei-ilyin/labs-template. (дата обращения: 21.09.2020.)

3. Соболь С.А. Методика преподавания дисциплин по теории алгоритмов с использованием образовательной платформы iRunner // Педагогика информатики. 2020. № 2. URL: http://pcs.bsu.by/ (дата обращения: 18.09.2020).