Решение Задач как Основа Обучения Программированию в Университете

Владимир Бондарев
кафедра программной инженерии
Харьковский национальный университет радиоэлектроники
Харьков, Украина
volodymyr.bondariev@nure.ua

Юлия Черепанова
кафедра программной инженерии
Харьковский национальный университет радиоэлектроники
Харьков, Украина
yulia.cherepanova@nure.ua

Solving Tasks as a Basis for Learning Programming in the University

Volodymyr Bondariev
Department of Software Ingineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
volodymyr.bondariev@nure.ua

Yulia Cherepanova
Department of Software Ingineering
Kharkiv National University of Radio Electronics
Kharkiv, Ukraine
yulia.cherepanova@nure.ua

***Аннотация*—С целью улучшения преподавания программирования предложена программная автоматической проверки задач. Предложенная система находится в эксплуатации и хорошо себя зарекомендовала.**

***Abstract*—In order to improve teaching of programming a software system of automatic testing for school assignment is offered. The proposed system is in operation and works well.**

Ключевые слова—обучение программированию; задача; автоматическая проверка решений; контроль знаний

Keywords—teaching of programming; a task; automatic testing; knowledge control

I. Введение

Программирование – относительно новое для человечества занятие, поэтому методика обучения ему еще не устоялась. Если задуматься, на что похоже начальное обучение программированию, придется признать, что скорее на математику или иностранный язык, чем историю или литературу. В самом деле, программист должен связно изложить в программе свои мысли, используя для их выражения весьма формальный аппарат, называемый языком программирования.

Известно, что в овладении математикой или иностранным языком ключевую роль играют упражнения, это верно и для программирования. Каким же образом заставить студента выполнить большое количество упражнений? Впрочем, почему заставить, ведь он должен упражняться добровольно, понимая, что это ключ к его будущей профессии. Да, есть и такие студенты, но, как показывает опыт, для большинства студентов данная мотивация не достаточна. Более реальным мотивом им представляется угроза получить плохую оценку, но и этого не достаточно для регулярных упражнений.

Ситуацию изменила бы обязательная проверка всех выполняемых заданий, но на это не хватает сил у преподавателя. Выход из тупика дает автоматизация проверки всех задаваемых упражнений, которые, без потери общности, можно свести к написанию фрагментов программ по заданной спецификации.

II. Обзор

В сети можно найти несколько версий задачников с автоматической проверкой решений. Главным образом, это системы автоматической проверки решений с архивами задач [1, 2] или без них [3] и опционально с возможностью участия или проведения соревнований по программированию [4, 5]. Задачи там абстрагированы от языка и унифицированы по форме – ввод из стандартного потока, вычисление результата, вывод результатов в стандартный поток. Всех их объединяет направленность

на соревнования по программированию, а не на обучение его основам. Есть также система, ориентированные на обучение [6, 7], но она не является сетевой и требует значительных усилий по развертыванию и настройке.

Задачник, представляемый ниже, отличается от всех упомянутых систем. Его особенностями являются тесная связь с другими составляющими учебного процесса (лекциями, практическими занятиями, самостоятельной работой, контролем знаний), легкость пополнения новыми задачами и разнообразие форм самих задач. Это могут быть задания на определение отдельных функций, классов или их членов, вычисление каких-то значений, исправление ошибок в программном коде, в общем, многое из того, что составляет повседневную работу программиста.

III. Архитектура

Задачник состоит из базы задач и сетевой службы, которая выполняет выдачу условий и проверку решений. Этой службой пользуются программные приложения, о которых будет сказано ниже.

Собственно задача состоит из словесного условия (рис. 1), авторского решения и тестового программного окружения, в которое помещено авторское решение (рис.2). Тестовое окружение вместе с авторским решением составляют законченную программу, которая может быть скомпилирована и выполнена. Авторское решение таково, что проходит тестовую проверку с положительным результатом (в противном случае задача не была бы принята в задачник).

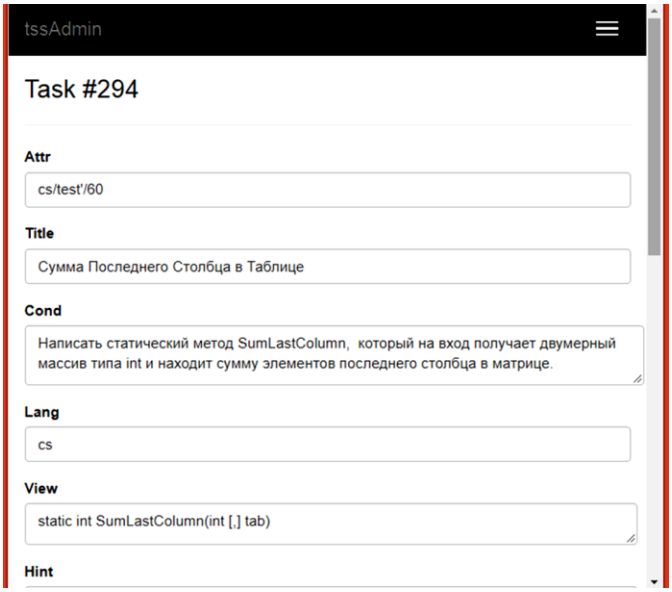

Рис. 1. Описание задачи.

Проверка студенческого решения состоит в том, что оно подставляется в тестовое окружение на место авторского, и полученная программа компилируется и выполняется. Если результат положительный, решение признается правильным, если результат отрицательный или программа не компилируется, решение отклоняется, а его автор получает сообщения об ошибках, которые выдал компилятор или тестовое окружение.

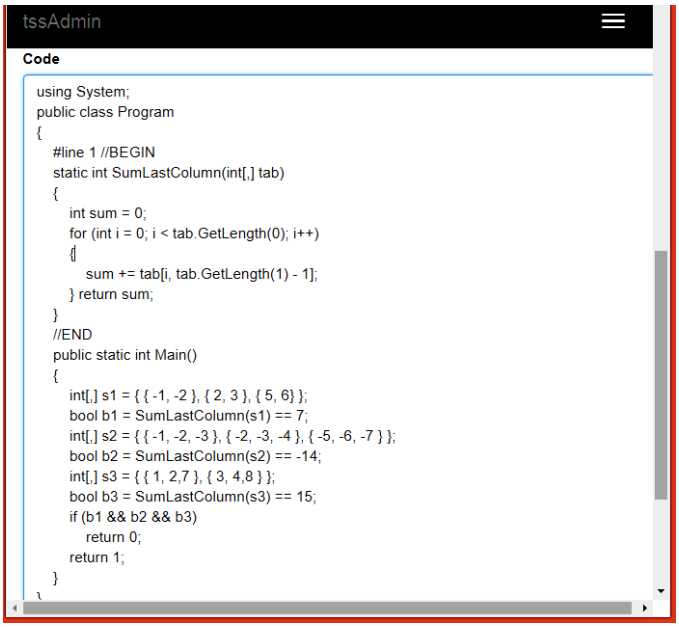

Рис. 2. Авторское решение и тестовое программное окружение задачи.

Студент получает задачи не от службы, а от прикладных программ, которые могут сопроводить базовые операции службы дополнительными действиями, такими как аутентификация пользователя, протоколирование его действий, контроль времени решения, переработку сообщений службы и т.п. Очевидными приложениями является программа публикации лекций со встроенными задачами и программа для проведения экзамена (рис.2), но могут быть и другие идеи для реализации, например, пополнение задачника силами студентов, вычисление различных рейтингов по результатам решений или иной статистики.

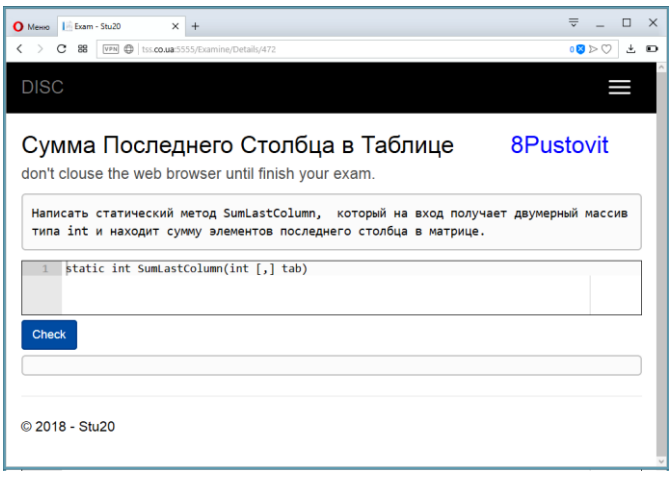

Рис. 3. Задача, предложенная студенту на экзамене.

Студенты и преподаватели являются зарегистрированным пользователями таких приложений,

таким образом, все их действия персонифицированы, хоть это не всегда очевидно.

IV. Задачи на экзамене

Благодаря автоматической проверке, контроль знаний превращается в предельно объективную процедуру, к тому же не обременительную для преподавателя. Он должен выбрать задачу, установить лимит времени и следить за тем, чтобы студенты работали самостоятельно. Такой способ проведение контрольных снимает любые вопросы к объективности оценок и улучшает климат в коллективе.

Преимущества такого экзамена будут обесценены, если нет гарантии, что задача решена самостоятельно. Довольно трудно пресечь все возможности коммуникации в эпоху мобильных устройств и интернета, но есть иные способы убедиться в оригинальности решения. Во-первых, это наличие истории решения в виде его промежуточных состояний и неудачных попыток. Во-вторых, можно сравнить данное решение с другими ("помощь" проще всего получить от товарищей по экзамену). Сравнение делается программой проведения экзамена и, если у решения обнаружен потенциальный прототип, это служит сигналом для проведения более глубокого анализа (рис. 4). Заметим, что сравнение можно делать с разной степенью "буквальности", вообще говоря, строится некий кэш решения и сравниваются кэши, а не сами решения.

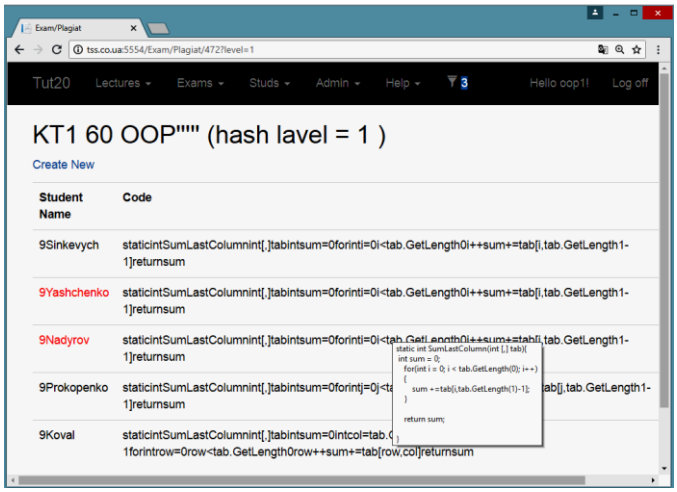

Рис. 4. Проверка решений на плагиат.

V. Задачи в лекциях

Задачи легко встраиваются в html-документ, каковым может быть конспект лекции, расположенный в сети. Прорабатывая лекции, студент может тут же проверить свое понимание учебного материала, решив контрольные задания (рис. 5). Сведения об удачных и неудачных попытках решения накапливаются в базе данных и в любой момент могут быть получены преподавателем.

Одним из немногих недостатков автоматической проверки решений является невозможность отличить хорошее решение от плохого (за исключением случаев, где это проявляется в производительности программы, здесь можно измерить время). Компенсировать этот недостаток можно дав возможность студентам обсуждать и оценивать решения других. Оценка решений подогревает интерес к задачам, что тоже немаловажно.

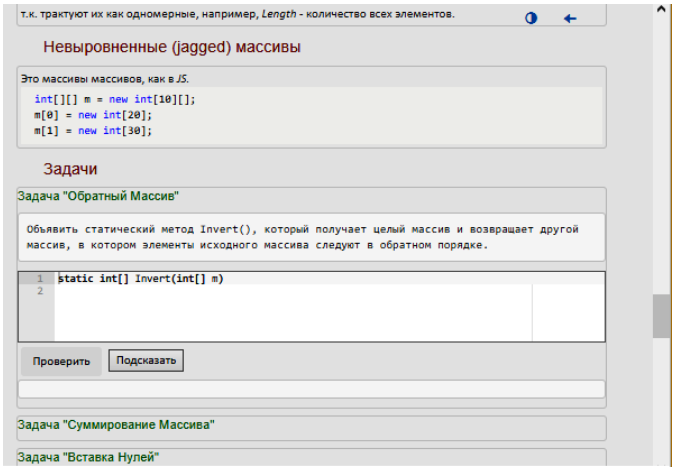

Рис. 5. Задачи в лекции.

Еще один вид активности – пополнение задачника самими студентами. Студентам можно предоставить возможность добавлять новые задачи, решать их, обсуждать и оценивать.

VI. Заключение

В процессе начального обучения программированию упражнения в написании программ занимают центральное место.

Противоречие между большим количеством заданий и ограниченными возможностями преподавателя может разрешить задачник с автоматической проверкой решений.

Для эффективного использования всех возможностей задачника необходимы программные приложения, расширяющие базовые функции задачника.

Литература References

[1] Online Judge – архив задач Вальядолидского университета в Испании. [Online]. Available: https://uva.onlinejudge.org/

[2] Timus Online Judge, problem archive with online judge system [Online]. Available: http://acm.timus.ru/

[3] ejudge - система для проведения различных мероприятий, в которых необходима автоматическая проверка программ. [Online]. Available: http://ejudge.ru/

[4] TopCoder – корпорация, проводящая соревнования по спортивному программированию. [Online]. Available: http://www.topcoder.com/

[5] CS Academy - образовательный проект в сфере программирования. [Online]. Available: https://csacademy.com/

[6] М. Э. Абрамян, С. С Михалкович. “Веб-среда разработки и обучения”, *Открытые системы. СУБД,* no. 10, с. 56–59, 2012

[7] M. E. Abramyan. Programming Taskbook. [Online]. Available: http://ptaskbook.com/ru/.