УДК 004.9

А.А. Буславский,
старший преподаватель кафедры естественнонаучных дисциплин и информационных технологий МОИРО, аспирант лаборатории математического и естественнонаучного образования НИО

ОРГАНИЗАЦИЯ ТЕСТИРОВАНИЯ РЕШЕНИЯ ОЛИМПИАДНОЙ ЗАДАЧИ ПО ИНФОРМАТИКЕ

Введение. От надежного функционирования определенных типов программного обеспечения может зависеть успех различных аспектов деятельности человека. Следовательно, на проектирование, разработку и тестирование расходуются значительные ресурсы (до 15 % от всех затрат) [1, с. 55]. Авторитетными организациями в области информатики в прошлом году был опубликован документ Computing Curricula 2013, который в значительной степени определяет содержание университетских учебных программ в области компьютерных наук на ближайшее время [2, с. 163]. Время, отводимое на изучение технологий тестирования, обычно составляет незначительное количество часов на четырехгодичный курс. Вследствие того, что олимпиадные задачи по информатике, как правило, проверяются системой тестов (наборов входных данных), то можно рассматривать составление тестов, с одной стороны, как инструмент проверки правильности решения (для членов жюри и разработчиков заданий), а с другой стороны, как инструмент обучения будущего специалиста информационных технологий умениям самопроверки собственного решения.

1. Профессиональное тестирование программного обеспечения. Процесс тестирования программного обеспечения можно отчасти назвать интуитивным, но в то же время в основе эффективного тестирования лежит стройная система.

Приведем рекомендации специалистов по организации первого цикла тестирования:

1) составление простого и наиболее очевидного теста;
2) составление заметок о том, что еще должно быть протестировано;
3) проверка допустимых значений;
4) тестирование на случайных наборах тестов;
5) подведение итогов о программе и ее недостатках [1].

Из огромного количества возможных тестов приходится отбирать реально выполнимую (в рамках отведенного на тестирование времени) часть, при этом все ошибки найти затруднительно. В известной книге по философии науки [3] Карл Поппер (Karl Popper) утверждает, что правильный подход к проверке научной теории заключается в поиске не подтверждающих, а опровергающих ее фактов – попытке доказать, что в ней есть ошибки. И чем более тщательное тестирование выдерживает выдвинутая теория, тем больше у нас уверенности в том, что она верна. Аналогичный подход используется при тестировании программного обеспечения. В конечном счете, большинство найденных ошибок исправляют, и качество программного продукта улучшается. Это и есть настоящая цель *тестирования*.

Продукт тестируют и исправляют практически на каждом этапе его жизненного цикла. При этом чем дальше продвигается работа, тем быстрее растет стоимость тестирования (на олимпиаде в качестве критерия можно рассматривать количество времени, затрачиваемого на проверку правильности работы программы).

Технология тестирования на этапе, когда программист составляет программу и сам ее тестирует, называется тестированием «стеклянного ящика» (glass box). Иногда ее еще называют тестированием «белого ящика» (white box) – в противоположность классическому понятию «черного ящика» (black box).

При тестировании «черного ящика» программа рассматривается как объект, внутренняя структура которого неизвестна. Подбирая тесты, специалист ищет интересные, с его точки зрения, входные данные и условия, которые могут привести к нестандартным (возможно, ошибочным) результатам. При тестировании «белого ящика» тесты разрабатываются на основании знания исходного текста программы.

Хороший тест должен удовлетворять следующим основным условиям:

- существует обоснованная вероятность выявления тестом ошибки;
- набор тестов не должен быть избыточным;
- тест должен быть наилучшим в своей категории;
- он не должен быть слишком простым или слишком сложным.

К основным методам тестирования программы можно отнести: автоматизацию (применение автоматизированных систем), анализ чувствительности (определение области, на которой небольшие изменения аргумента вызывают значительное изменение результата) и случайный ввод (позволяет выявить непредвиденные ошибки).

2. Тестирование олимпиадной задачи по информатике. Олимпиадное задание по информатике для школьника предполагает написание алгоритма и реализацию его на одном из допущенных языков программирования. Задание может состоять из названия, текста (который может включать пояснения и пробные тесты), обзорного листа (оценка частичных решений). Жюри также обычно получает набор тестов для проверки, авторское решение и пояснение к решению.

В настоящий момент в международной олимпиаде по информатике (и олимпиаде по информатике Республики Беларусь, проводимой в четыре этапа) используются 3 типа заданий: классическое; использование библиотеки; с открытыми тестами.

Классическое задание предполагает наличие одного или нескольких входных файлов, на основании которых программа участника должна сформировать выходной файл, придерживаясь ограничений на формат вывода, использование времени и памяти.

Задача с использованием библиотеки предполагает использование подпрограмм этой библиотеки. Участник знает только спецификацию подпрограммы (описание и назначение) и, возможно, имен некоторых глобальных переменных (констант). Данный тип заданий не предполагает прямую работу с файлами – она осуществляется подпрограммами библиотеки.

Задача с открытыми тестами отличается от вышеописанных тем, что участник получает набор входных тестов, на которые любым способом должен сформировать набор выходных тестов. Некоторые тесты можно вычислить вручную, для получения остальных необходимо писать программы. Участник сдает только выходные файлы.

При оценивании задачи вычисляется сумма баллов пройденных тестов. Оценка участника за тур состоит из суммы оценок его задач.

Для тестирования решения участников используется ручное (полуавтоматическое) тестирование решения членами жюри (чаще всего на первом и втором этапе республиканской олимпиады по информатике) или система автоматической проверки (как правило, на третьем и / или четвертом, то есть заключительном, этапах). Автоматизированная система обычно используется в том случае, если все компьютеры участников можно объединить в сеть, обеспечив необходимые уровни защиты.

Умение составлять тесты важно, в первую очередь, авторам олимпиадных заданий (для школьников или студентов), учителям (для проверки решения ученика в процессе тренировки) и школьникам (для самопроверки).

Рассмотрим профессиональное тестирование программного обеспечения в рамках возможности использования методов тестирования для олимпиадных заданий по информатике.

Прежде всего, отметим, что умение проверять правильность решения задачи отсутствует в программе предмета «Информатика» для общеобразовательных учреждений. Учителя могут попытаться выработать это умение в процессе решения задач по программированию. Времени на выработку умения не хватает, и в лучшем случае формируются только реакция на сообщения компиляции и умение использовать инструменты пошаговой отладки. Правильность же решения задачи, как было сказано выше, – это не только «соответствие спецификации», то есть правильность программирования заданного алгоритма, но и практическая проверка самого алгоритма.

При составлении тестов можно использовать советы, приводимые для профессионалов. Продемонстрируем это на простейшем примере, с которым не может справиться большинство начинающих участников олимпиады по информатике.

Задача 1. Вывести сумму двух целых чисел, не превышающих по модулю **longint**.

Входной файл (input.txt): x y. Выходной файл (output.txt): x+y.

Пример: Входной файл: 2 2. Выходной файл: 4.

Анализ тестирования задачи. Вначале продемонстрируем использование «белого ящика», то есть проверка человеком исходного текста программы. Пусть решение ученика было следующим:

```
Program z1;
  var a,b:integer; f1,f2:text;
  Begin
  assign(f1,input.txt); reset(f1);
  assign(f2,'output.txt'); rewrite(f2);
  read(f1,a,b); writeln(f2,a+b);
  close(f1);close(f2);
End.
```

Первое, что необходимо сделать, – свериться со спецификацией. Проверяем, что программа работает с текстовыми файлами, правильность их именования, корректность открытия / закрытия и правильность использования операторов ввода / вывода для работы с файлами. Кроме того, проверяем форматы ввода вывода.

Варианты ошибок: `readln(f1,a)`; `readln(f1,b)`; (в этом случае программа прочитает только одно число из правильного входного файла, что приведет к ошибке); `writeln(f2,a+b)`; (использование `writeln` вместо `write` создаст пустую вторую строку в выходном файле. Это в некоторых случаях может дать несовпадение с авторским ответом, вплоть до нуля баллов за решенную, казалось бы, задачу.

Второе – пробный тест. Вводим тот, что приведен в задаче, и проверяем полученный вариант (выполняем пошаговую отладку мысленно, или на бумаге). Также проверяем несколько случайных тестов, например: 2 2; 1 3; 4 5; -1 -1.

Третье – соответствие спецификации: программа действительно получает и выводит сумму двух чисел.

Четвертое – использование структур и типов данных. Программа не превышает лимит предоставленной памяти, но использование типа **integer** вызывает вопрос о соответствии граничным условиям (в некоторых версиях значение типа **integer** не превосходит по модулю 32767. Пример теста: 50000 50000.

Пятое – проверка граничных условий. Даже если каждое из чисел помещается в типе, поместится ли сумма? Пример теста: 2000000000 2000000000.

Шестое – создание системы тестов на основании предварительного тестирования. И если для «белого ящика» все уже достаточно понятно, то для «черного ящика» нужно выяснить, не использовал ли ученик беззнаковые переменные, корректно ли он работает с отрицательными значениями (возможно, использует модуль), и т. д. Понятно, что все возможные логические ошибки система тестов не обнаружит, но задачей является определить наиболее вероятные ошибки.

Рассмотрим пример разработки группы тестов для «черного ящика»:

Задача 2. Даны координаты центра окружности, ее радиус, а также координаты противоположных вершин прямоугольника со сторонами, параллельными осям координат (все числа являются целыми, не превышающими по модулю 1000000000). Найти количество общих точек заданных фигур.

Входной файл (geom.in): **X Y R X1 Y1 X2 Y2**. Выходной файл (geom.out): **N**.

Пример: Входной файл: **0 0 10 10 10 1 1**. Выходной файл: **2**.

Краткий анализ тестирования задачи. Определение соответствия спецификации работы с файлами будет осуществлено в ходе тестирования. Если участник не работает с файлами или использовал неправильные имена файлов, то выходных файлов либо не будет, либо они будут заполнены неверными данными. Распространенным случаем будет содержание «0» в выходном файле, поэтому нужно уменьшить количество таких тестов, где «0» является правильным ответом. Нарушение формата вывода (например, участник хранил ответ в переменной вещественного типа и вывел число как вещественное, например, **2.000**) может быть осуществлено проверяющей программой (чекером, *checker* – англ.).

Пробный тест дается для самопроверки, но это не запрещает использовать его и в качестве одного из оценочных. Для олимпиады, где используется автоматизированная система, участник может проверить свою программу с помощью системы как на пробном тесте, так и на своем.

Структуры и типы данных, используемые в данной задаче, достаточно простые. Тем не менее, участник мог невнимательно прочитать условие и использовать недостаточный числовой тип (например, byte). Этот случай необходимо, конечно, проверить, но большинство тестов не должно выходить за минимально необходимые пределы, чтобы найти другие ошибки.

Самый интересный блок в этой задаче – определение граничных условий. Существует 9 возможных ответов на задачу (числа от 0 до 8), причем 0 может быть как в случае, когда фигуры лежат отдельно, так и одна внутри другой. Таким образом, необходимо составить как минимум 9 тестов с разными ответами. Кроме того, важно учитывать как точки пересечения, так и точки касания прямоугольника и окружности, так как эти точки могут определяться различными алгоритмами.

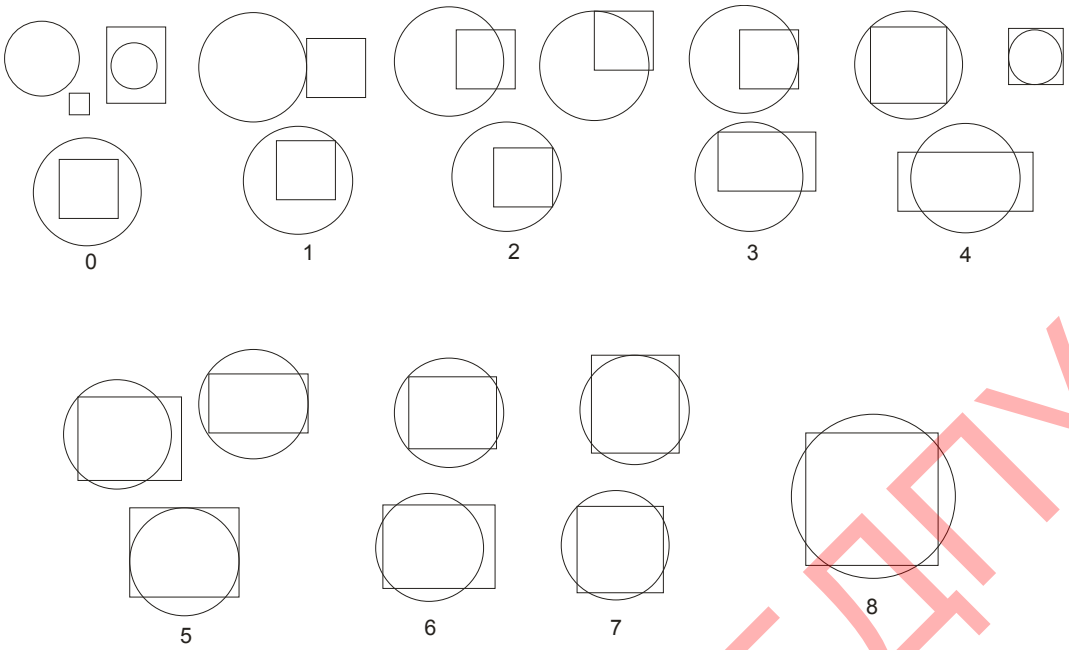

Рисунок – Варианты тестов задачи 2

К сожалению, невозможно проверить все случаи, но можно выбрать наиболее типичные. Кроме того, некоторые из случаев, отображенные на рисунке, сложно реализовать входными данными с целыми числами. Помимо просто проверки работоспособности программы разработчик тестов для олимпиадной задачи по информатике должен учитывать возможность решения различными способами и дифференцировать тесты по оптимальности используемого метода (в данной задаче это несущественно). Тесты могут иметь как одинаковую стоимость, так и различаться. Изменение стоимости тестов может привести к другим результатам оценки задачи, меняя ее сложность (с точки зрения процента решения).

Участник знает, что его программа проверяется системой тестов, и может так построить нахождение решения, чтобы прошло наибольшее число тестов. Например, если вариантов ответа немного (например, «yes» или «no»), то можно всегда выводить один и тот же ответ, ожидая прохождения около половины тестов. В таком случае тестировщик обычно создает группу тестов и засчитывает только прохождение всех в группе. Другой пример – выдача в качестве ответа случайного значения. В этом случае тестировщик может провести проверку каждого теста несколько раз и взять худший результат в качестве ответа.

На основании такого тестирования можно оценить задачу. Однако для ученика (и его учителя) на олимпиаде важна не только полученная оценка задачи, важным является также определение наличия ошибок в его программе. И можно провести анализ непройденных тестов, исправляя их в текущей программе и запоминая для подобных случаев в будущем. Кроме того, регулярно анализируя грамотно составленные тесты, можно научиться составлять тесты самому.

Автор задачи, составляя тесты, выделяет ключевые особенности используемого для решения алгоритма. Так, если при использовании алгоритма быстрой сортировки (сортировки Хоара) [4] эталонный элемент выбирается как средний по положению, то можно подобрать тест, делящий числа на две группы, одна из которых состоит из единственного элемента. Это ухудшает оценку алгоритма до O(n2), то есть одна из лучших сортировок сравнениями работает не лучше, чем простейшая сортировка обменом («пузырек») [4].

Заключение. В результате проведенного исследования установлено, что для проверки правильности решения участника олимпиады по информатике используется метод тестирования, то есть проверка работы на различных наборах входных данных. При формировании тестовых наборов могут быть использованы методы, применяемые специалистами в сфере тестирования программного обеспечения. Следовательно, разработка олимпиадных заданий и подготовка учащихся к олимпи-

аде по информатике способствуют получению специфических знаний, умений и навыков специалиста информационных технологий в сфере тестирования программного обеспечения.

Результаты исследования могут быть использованы авторами задач для олимпиад по информатике, учителями, разработчиками курса предмета «Информатика», спецкурсов и факультативов по информационным технологиям, авторами учебной и методической литературы по направлению, для организации совместного процесса подготовки участника олимпиады и получения знаний, умений и навыков специалиста по информационным технологиям.

Литература

1. *Канер, Сэм* Тестирование программного обеспечения. Фундаментальные концепции менеджмента бизнес-приложений: пер. с англ. / Сэм Канер, Джек Фолк, ЕнгКекНгуен. – К.: Изд-во «ДиаСофт», 2001. – 544 с.
2. Computer Science Curricula 2013 (CS2013) Ironman Draft (Version 1.0) [Electronic resource] / The Joint Task Force on Computing Curricula, Association for Computing Machinery, IEEE-Computer Society, CS2013 Steering Committee. – Mode of access: http://ai.stanford.edu/users/sahami/CS2013/ironman-draft/cs2013-ironman-v1.0.pdf. – Date of access: 01.03.2013.
3. *Поппер, К.* Логика и рост научного знания / К. Поппер. – М.: Прогресс, 1983.
4. *Буславский, А.А.* Начальный уровень обучения программированию на языке Pascal: учеб.-метод. пособие / А.А. Буславский. – Минск: Минский областной ИПКиПРРиСО, 2003. – 72 с.

Summary

The examination of software is the main way of checking of its functioning at the stage of working out. The same method is applied to determine the correctness of participant's solution at the Olimpiad in Informatics. The article substantiates the hypothesis that student's preparation for competitions assists to obtain specific knowledge and expert skills in the sphere of information and software.

Поступила в редакцию 19.02.2014 г.