Введение

Учебные планы программистских факультетов большинства вузов подразумевают, что студенты-первокурсники уже умеют программировать. Однако тот курс программирования, который входит в школьную программу, недостаточен. В частности, в нем почти не уделяется внимания таким важным вопросам, как тестирование и отладка. Профессионалы знают, что затраты на тестирование и отладку оцениваются в 50–60% всех затрат на разработку программы. Но для подавляющего большинства школьников, да и многих студентов «написать программу» означает написать некий текст на языке программирования и, в лучшем случае, добиться того, чтобы в нем не было ошибок трансляции. О проверке соответствия программы поставленной задаче, поиске и исправлении смысловых ошибок, как правило, речь даже не заходит. Причиной этого в большой степени является отсутствие наглядной и доходчивой методики тестирования и отладки программ.

В данной книге изложена методика тестирования учебных программ, основанная на классических научных исследованиях и опыте преподавания начального курса программирования в старших классах средней школы и младших курсах университета. Подробно рассматриваются отладочные средства популярной системы Турбо Паскаль. В главах 14 и 15 описываются статистические модели оценки количества ошибок в программе и количества тестов, необходимых для их обнаружения. Изучение этого материала требует определенной математической подготовки. Данные главы отмечены звездочкой.

Классик жанра Гленфорд Майерс свою легендарную книгу «Искусство тестирования программ» начал с того, что предложил читателям сформировать набор тестов для проверки простенькой программы. Программа читала 3 целых числа (у Майерса — с перфокарт, мы скажем — из файла; впрочем, кто не умеет вводить числа из файла, может вводить их с терминала), рассматривала их как длины сторон треугольника и печатала сообщение о том, является ли этот треугольник разносторонним, равносторонним или равнобедренным. Далее набор тестов, предложенный читателем, надо было сравнить с набором, предложенным самим Майерсом.

Мы также используем майерсову задачу, но порядок работы несколько изменим. Читателю предлагается составить набор тестов для задачи про треугольники. Но обсуждать полученный набор немедленно мы не будем. Вместо этого мы еще несколько раз вернемся к этой задаче по ходу изучения темы с тем, чтобы читатель мог на практике применить получаемые знания и оценить свой прогресс в этой области. И только после этого мы обсудим полученный результат.

Итак, перед дальнейшим чтением вам предлагается самостоятельно составить набор тестов для вышеуказанной программы.

Готово? Тогда — продолжим.

Глава 1

В каком случае программа содержит ошибку?

Понятия тестирования и отладки связаны с процессом поиска и исправления ошибок в программе. Поэтому первый вопрос, на который надо ответить, будет звучать так: в каком случае в программе есть ошибка?

Программа содержит ошибку, если она ведет себя неразумно с точки зрения пользователя.

Это утверждение повергает новичков в замешательство: откуда же я знаю, какое поведение программы пользователь сочтет разумным? Но если вы не знаете, какое поведение программы разумно с точки зрения вашего заказчика, значит, вы не понимаете, какую задачу решаете. Как можно писать программу, не понимая, что она должна делать?

Как определить разумность поведения программы?

Во-первых, естественно, программа должна быть верна синтаксически, т. е. при ее трансляции не должно быть ошибок. Текст, содержащий синтаксические ошибки, вообще не имеет права называться программой. Такая «программа» вообще никак себя не ведет.

Во-вторых, программа должна правильно решать поставленную перед ней задачу. То есть при вводе в нее корректных исходных данных она должна выдавать правильный результат. Какой именно результат считать правильным, надо уточнить у заказчика. Например, если вам заказывают программу для вычисления квадратного корня, то должна ли она выдавать оба корня (положительный и отрицательный) или только арифметический? Корень из нуля — это ±0 или просто 0? Какова требуемая точность? Она всегда должна быть одна и та же или может меняться? Если меняться, то каким образом и по чьей инициативе? Надо ли выявлять периодические дроби? Как программа должна реагировать на отрицательное подкоренное выражение? А на предложение извлечь корень из a^2? Вы можете предложить свои ответы на все эти вопросы. Но в данном случае важно не то, что думаете по этому поводу вы, а то, что думает по этому поводу ваш заказчик. Ваша задача — выявить и сформулировать все эти вопросы, а не придумывать на них свои собственные ответы. Для выявления и формулировки вопросов полезно не хвататься за клавиатуру сразу же, как толь-

ко вам дадут задание на разработку программы, а сначала немножко подумать. Все эти вопросы все равно возникнут в процессе разработки программы. Но если их не оговорить заранее, ответы на них вы будете придумывать сами, и нет никакой гарантии, что они покажутся заказчику разумными.

В-третьих, программа не должна делать ничего лишнего. При вычислении квадратного корня не надо менять настройки операционной системы и исполнять вашу любимую мелодию.

В-четвертых, результат должен быть получен через разумное время при разумных затратах других ресурсов. Если программа при вычислении квадратного корня из числа выдает десятистраничную распечатку, в ней, наверное, что-то не так.

И наконец, в-пятых, программа должна разумно реагировать на ввод некорректных входных данных и на непредусмотренные заранее ситуации. Например, если программа вычисляет квадратный корень, а пользователь вместо подкоренного выражения ввел свою фамилию, то программа должна распознать, что введенная строка — не число[1], и сообщить об этом пользователю на его родном языке, а не заканчивать работу аварийно. Если вы разработали программу для управления аэропортом, рассчитанную на то, что в небе могут быть одновременно не более 20 самолетов, а в какой-то момент их оказалось 21, ваша программа должна отреагировать на это неким разумным образом. Например, сообщить о чрезвычайной ситуации диспетчеру и экипажу и передать этот борт диспетчеру для ручного ведения. Вариант, когда появление 21-го самолета приведет к тому, что программа потеряет один из ранее ведомых самолетов или просто отключится, совершенно недопустим.

В поддержку данного выше утверждения об ошибках в программе в [2] упоминается следующая история. При разработке системы противоракетной обороны США военные заказчики потребовали от программистов, чтобы система подавала сигнал тревоги, если обнаружит в небе враждебный летательный объект. Программисты спросили, какой объект считать враждебным. Военные ответили: любой, который не будет опознан как дружественный. Программисты добросовестно заложили в систему полученные от военных правила распознавания дружественных летательных объектов. В ночь испытания системы над горизонтом взошла Луна... Была ли в данном случае выдача сигнала к началу III Мировой войны верным поведением программы, или в ней все-таки была ошибка?

---

[1] В Турбо Паскале для преобразования последовательности литер, представляющей собой изображение числа, в число (и проверки правильности записи числа) используется процедура `Val`. Для организации собственной диагностики любое значение следует вводить как строковое и преобразовывать в число с помощью процедуры `Val`.

Глава 2

Минимальные требования к программе: функциональность и удобство использования

Минимальные требования к любой программе — это обеспечение требуемой функциональности (реализация требуемых функций) и удобство эксплуатации. Именно они должны быть проверены в первую очередь.

Почти вся оставшаяся часть книги будет посвящена тестированию функциональности. Удобство взаимодействия человека с программой останется вне нашего внимания. Относительно него ограничимся только одним замечанием. Лозунг современного интерфейса: «Не пользователь должен приспосабливаться к программе, а программа к пользователю». Даже самая простая программа должна обеспечить пользователю некоторый уровень комфорта. Например, совершенно недопустима ситуация, когда после запуска программы перед пользователем возникает пустой экран с мигающим курсором и больше ничего. Пользователю предлагается самому догадаться, что за программу он запустил и что он должен делать дальше. Как минимум, программа должна сообщить о своей функциональности и объяснить свое поведение (выдать осмысленное приглашение к вводу или сообщить, почему возникла пауза).

Глава 3

Понятия тестирования и отладки

Тестирование — это выполнение программы с целью обнаружения факта наличия в программе ошибки.

Отладка — определение места ошибки и внесение исправлений в программу.

В русском языке словосочетание «обнаружить ошибку» может иметь, по крайней мере, два смысла: «обнаружить факт наличия ошибки» и «обнаружить место, где допущена ошибка». Под тестированием понимается обнаружение ошибки в первом смысле, под отладкой — во втором.

Как правило, эти два процесса тесно взаимосвязаны, но могут быть и разделены. Например, в процессе приемки системы заказчик занимается только тестированием — поиском несоответствий между заданием на разработку программы (спецификацией программы) и разработанной программой. Если его поиски увенчаются успехом, программисту придется заниматься отладкой (определением места допущенных ошибок и исправлением программы).

Цель тестирования — обнаружение ошибок в программе.

Отметим важный психологический момент: цель тестирования — не доказать правильность программы, а обнаружить в ней ошибки! Если вы ставите себе задачей показать, что программа правильная и ошибок не содержит, то (на подсознательном уровне) и тесты будете подбирать такие, которые ошибок в программе не обнаружат.

Отсюда следующее неожиданное рассуждение. Зададим вопрос: какой тест считать удачным? Если цель тестирования — найти в программе ошибку, то удачным должен считаться тест, который обнаруживает наличие в ней ошибки. Это положение противоречит интуитивному представлению о тестировании и требует сознательного психологического настроя.