операций, и лишь небольшая его часть непосредственно касалась вывода символа на экран. Если бы интерфейс процедуры позволял указывать в качестве аргумента не символ, а строку символов, тогда печать целой строки выполнялась бы лишь немного медленнее, чем печать одного символа. Результатом такого неудачного проектирования стала ужасающе медленная скорость вывода на экран всех компьютеров, работающих под управлением DOS. Этот же принцип касается и проектирования криптографических систем. Убедитесь, что работа может выполняться сразу большими порциями. Затем оптимизируйте только те части программы, влияние скорости работы которых на общую производительность системы может быть *измерено* и является существенным.

9.4.3 Утверждения

Утверждения (assertions) — это хорошее средство, с помощью которого можно повысить качество кода[5].

Реализуя криптографический код, необходимо подходить к нему с точки зрения "профессиональной паранойи". Каждый модуль не доверяет другим модулям и всегда проверяет достоверность параметров, накладывает ограничения на вызывающую последовательность и отказывается выполнять небезопасные операции. В большинстве случаев это описывается явными утверждениями. Если спецификации модуля утверждают, что перед использованием объекта его необходимо инициализировать, тогда попытка использования объекта до его инициализации приведет к возникновению ошибки утверждения. Ошибки утверждения всегда должны приводить к аварийному завершению программы с выдачей подробного сообщения о том, какое из утверждений было нарушено и почему.

Общее правило выглядит следующим образом: каждый раз, когда вы выполняете какую-либо значимую проверку внутренней согласованности системы, добавьте к ней утверждение. Постарайтесь перехватить как можно больше ошибок (и собственных, и допущенных другими программистами). Ошибка, перехваченная с помощью утверждения, не приведет к появлению "бреши" в системе безопасности.

Некоторые программисты реализуют проверку утверждений в процессе разработки программного обеспечения, однако отключают ее в готовом продукте. Хотелось бы знать, кто это выдумал. Что бы вы сказали об атомной электростанции, операторы которой тренируются работать с реакторами при включенных системах безопасности, а затем отключают их на настоящем

[5]Мы знаем, что наша книга потихоньку превращается в урок программирования, но что поделать? Мы постоянно вынуждены повторять эти, казалось бы, очевидные вещи программистам, с которыми работаем.

реакторе? Или о парашютисте, который одевает запасной парашют на тренировках, но снимает его, когда выпрыгивает из самолета? Зачем кому-то вообще понадобилось отключать проверку утверждений в готовом продукте — ведь это, по сути, единственное место, где она нужна? Если в процессе функционирования реальной системы произойдет нарушение утверждения, мы всего-навсего получим ошибку программирования. Игнорирование ошибки, в свою очередь, может привести (и, скорее всего, приведет) к выдаче неверного ответа, потому что по крайней мере одно предположение, сделанное кодом, будет неправильным. Выдача неверных ответов — это, пожалуй, худшее, что может сделать программа. Гораздо лучше проинформировать пользователя о возникновении ошибки программирования, чтобы он не доверял ошибочным результатам, выданным программой. Никогда не отключайте проверку ошибок!

9.4.4 Переполнение буфера

Современной IT-индустрии должно быть стыдно за появление в нашей книге раздела с таким заголовком. Проблемы переполнения буферов преследуют нас на протяжении уже 40 лет. Все это время в мире существовали и решения для борьбы с ними. Некоторые из ранних языков программирования высокого уровня, например Algol 60, полностью решали эту проблему путем введения обязательной проверки границ массива. К сожалению, даже несмотря на наличие массы поистине замечательных решений, переполнение буферов до сих пор является причиной более половины всех проблем безопасности, возникающих в Internet. Устранять же эти проблемы никто не собирается. Мы считаем это преступной халатностью. Что бы мы подумали, если бы производитель автомобилей сделал бензобак из оберточной бумаги? Конечно же, при определенной доле везения машина могла бы прекрасно ездить и с таким бензобаком, но руководство компании-производителя все равно угодило бы за решетку. Между тем целые отрасли IT-индустрии ведут себя так, будто не являются ответственными за последствия своих действий. (Возможно, наши юристы разрешают им свободно отказываться от своих обязательств, что было бы неприемлемо в любой другой области.) Наблюдая подобное отношение к программному обеспечению, мы часто задумываемся: а стоит ли вообще пытаться внедрять что-нибудь такое сложное, как криптография?

К сожалению, мы не в состоянии изменить текущее положение дел. Мы можем лишь посоветовать вам, как написать хороший криптографический код. Откажитесь от языков программирования, которые допускают переполнение буфера. В частности, не используйте C или C++. И никогда не отключайте проверку границ массива, какой бы язык вы не использовали. Это,