Статья опубликована в методической газете для учителей информатики «Информатика». 2008. № 19, с. 20 – 28.

Об опыте участия в командных соревнованиях по программированию формата *ACM ICPC*

И. Р. Акишев
Санкт-Петербургский государственный университет информационных технологий, механики и оптики
Золотой призер командного чемпионата мира по программированию среди студентов *ACM ICPC* 2007 года

В книге *Скиена С., Ревилла М.* Олимпиадные задачи по программированию. Руководство по подготовке к соревнованиям. М.: Кудиц-Образ, 2005 авторы обращают внимание читателей на важность обмена опытом *любых участников финалов* чемпионата мира формата *ACM ICPC*. Тем более интересен опыт автора, участника команды, занявшей третье место (золотая медаль) на этих соревнованиях в 2007 году.
А. А. Шалыто, докт. техн. наук, профессор

Предисловие

В марте 2007 года в Токио прошел финал XXXI командного чемпионата мира по программированию среди студентов *ACM ICPC (International Collegiate Programming Contest)*. Это соревнование по праву считается одним из самых престижных соревнований по программированию в мире. Проводят его такие всемирно известные организации, как *ACM (Association for Computing Machinery)*, *UPE* – международное почетное общество в области Computer Science, а главным спонсором выступает компания *IBM* – мастодонт в области компьютерных технологий. Ежегодно более шести тысяч студенческих команд из 1700 университетов со всего света соревнуются за право попасть в финал чемпионата, где около сотни самых лучших из них будут биться за почетный титул абсолютных чемпионов мира.

В 2007 году среди команд, участвовавших в финале этих соревнований, была команда Санкт-Петербургского государственного университета информационных технологий, механики и оптики (СПбГУ ИТМО). Тогда, в 2007 году, ей удалось занять почетное третье место и завоевать золотую медаль. Автору этой статьи посчастливилось быть одним из членов этой команды.

Годом ранее, наша команда в том же составе неудачно выступила на юбилейном XXX финале чемпионата мира, проходившем в Сан-Антонио (Техас), решив всего три задачи из десяти предложенных. Тогда мы не завоевали никаких медалей или дипломов и оказались в итоговой таблице результатов на местах с 19 по 38 (вместе с обширной группой из еще 19 таких же не самых удачливых команд).

За годы тренировок и участия в различных соревнованиях по программированию (начиная со школьных), пройдя через множество побед и поражений, мне удалось накопить некоторый опыт, который мог бы быть полезен тем, кто планирует участвовать в командных соревнованиях по программированию, и которым я бы хотел поделиться.

Для кого эта статья

Эта статья, в первую очередь, предназначена для школьников и студентов, увлекающихся спортивным программированием, которые уже имеют опыт участия в личных соревнованиях и собираются принимать участие в командных турнирах (или уже это делают). Предполагается, что читатель имеет представление о том, как решать различные олимпиадные задачи.

В этой статье я буду крайне мало говорить о том, какими бывают олимпиадные задачи и как их решать. Для того чтобы быстро, эффективно и правильно решать задачи требуется быть хорошим программистом и математиком, знать множество алгоритмов и структур данных из области *computer science*, иметь большой практический опыт – много и регулярно тренироваться. Более того, в конце концов, умение решать задачи – это в какой-то степени талант, дар, ведь на самом деле решение любой сложной олимпиадной задачи – это всегда творческий процесс. Нельзя придумать никакого общего алгоритма, который помог бы вам гарантированно найти правильное решение произвольной задачи.

Так что же все-таки делать, чтобы хорошо решать задачи? Как сказал Конфуций: «Чтобы научиться плавать – нужно плавать». Тренируйтесь, участвуйте в различных олимпиадах, посещайте кружки, читайте книжки, изучайте новые алгоритмы, слушайте советы ваших тренеров и обменивайтесь опытом с вашими друзьями, которые, как и вы, увлекаются спортивным программированием **(**если они, правда, захотят вас слушать :-**))**.

Если вы пока что не можете придумать решения каких-то сложных задач сами, то узнавайте их, разбирайте и тренируйтесь в их реализации. Накапливайте опыт, пытайтесь выделять в решениях разных известных вам задач схожие идеи, приемы и алгоритмы для того, чтобы применять их и далее. Учитесь на своих ошибках. Эту фразу, в частности, можно понимать буквально: если вы в процессе решения какой-то задачи допустили ошибку, неважно в самой идее решения или в ее реализации, не забывайте про нее после исправления, а анализируйте, находите ее причины, запоминайте и старайтесь избежать появления подобных ошибок в решениях других задач.

Желающих получить какие-то более конкретные и подробные идеи и советы по поводу решения олимпиадных задач я отсылаю к статье Сергея Оршанского, победителя чемпионата мира *ACM ICPC* 2004 года и золотого призера чемпионата мира *ACM ICPC* 2005 года [1].

Далее я постараюсь сконцентрироваться на основных особенностях командных соревнований по программированию и на их отличиях от личных соревнований и сформулирую некоторые советы и рекомендации, поделюсь своими соображениями по поводу командной тактики и психологии. Надеюсь, эти советы помогут вашей команде стать более организованной и слаженной и, как следствие, показывать высокие и стабильные результаты на различных соревнованиях. Выполнение некоторых советов (в отличие от советов по поводу решения задач) не требует особых усилий. Часть из них даже довольно проста и очевидна. Однако, они могут помочь вам достичь высоких результатов в командных турнирах иногда не в меньшей степени, чем собственно умение хорошо решать задачи. Особенно это касается молодых и неопытных команд.

Отличия командных соревнований от личных

Традиционно сложилось так, что в нашей стране, как, впрочем, и во всем мире, более важную роль среди олимпиад для школьников играют личные соревнования (формата *IOI*), а для студентов – наоборот, командные (формата *ACM ICPC*). Конечно же, бывают и командные соревнования школьников (например, всероссийская командная олимпиада школьников, проводимая ежегодно в Санкт-Петербурге), и различные личные соревнования студентов (*TopCoder*, *Test-The-Best* и т.д.), но основная тенденция именно такова. Такая ситуация вовсе не случайна. С одной стороны, школьники – это еще дети. Они, как правило,

более индивидуальны, эмоциональны, и, как следствие, в меньшей степени способны организовано взаимодействовать друг с другом, работать в команде. Олимпиады по программированию должны давать талантливым школьникам возможность показать свои навыки решения задач, способности к аналитическому и в то же время творческому, нестандартному мышлению, раскрыть свой собственный потенциал.

Основные цели соревнований среди студентов уже несколько иные. Студенты – более взрослые люди, у них больше как практического опыта и фундаментальных знаний, так и способностей к самоорганизации и кооперированию. Поэтому в командных соревнованиях уровень сложности задач уже на порядок выше, их число и длительность соревнования больше, а требования к решениям – жестче. В таких условиях один в поле – уже «не воин», и, для того чтобы добиться результата, необходима слаженная командная работа. Кроме того, когда команда студентов выступает на международных соревнованиях, она защищает не только честь своей страны и своего города, но и в не меньшей степени своего университета. И тут уже соревнуются не просто отдельные одаренные личности, а различные школы олимпиадного программирования с многолетним опытом, сплоченные команды профессионалов [2, 3].

Эти отличия, с одной стороны, делают командные соревнования студентов более интересными, чем школьников, а с другой – несомненно, более сложными. Давайте остановимся поподробнее на основных различиях олимпиад форматов *ICPC* и *IOI* и вытекающих из них наблюдений.

Командное участие. Безусловно, самым главным отличием командных соревнований от личных является сам факт того, что участники соревнований формата *ICPC* решают одни и те же задачи вместе, в команде из трех человек, а не по одиночке. Это во многом усложняет командные соревнования и дает участникам огромный простор при выборе тактики и стратегии.

Когда на олимпиаде выступает один участник, то для него не так важно в каком порядке читать условия задач. Он сам, исходя из своих индивидуальных способностей, принимает решение о том, какую задачу он будет решать первой, а какую отложит на потом (зная, что один тип задач ему дается легче, чем другой), стоит ли еще раз протестировать написанную только что задачу, или можно уже взяться за новую, искать ли ошибку в решении, запуская его при помощи отладчика, или внимательно перечитывая исходный код программы. Он будет называть переменные в коде так, как ему привычнее, ведь никто кроме него уже не должен будет вникать в этот код. И так далее.

В команде появляются такие варианты работы, как парное программирование. Код, который пишет один из участников команды, могут читать и проверять остальные, поэтому стоит делать его читаемым и понятным. Здесь придется задумываться над тем, кто будет писать решение той или иной задачи за компьютером. Если есть несколько участников, которые знают его идею, или, если разные участники хотят запрограммировать разные решения, то придется решать, кто из них будет первым, а кто подождет. Сам процесс «придумывания» решений здесь уже можно (и нужно) распараллелить, распределив задачи между участниками. Однако каким образом сделать это оптимально? Одним словом, здесь все уже становится намного сложнее.

Кроме собственно навыков решения задач важную роль начинают играть способности участников команды работать друг с другом вместе, умение разделять задачи между собой и распределять их решение по времени тура так, чтобы максимально использовать все доступные ресурсы (компьютер, свои *светлые* головы, монитор соревнований, и все остальное) для достижения хорошего результата. Особое значение имеет и сам климат в команде, взаимопонимание участников, способность быстро и без долгих споров принимать решения коллективно, умение, с одной стороны, без лишних эмоций относиться к ошибкам своих «сокомандников», и в то же время учитывать и воспринимать рационально их возможные замечания и советы, а так же замечания тренеров.

Число и уровень задач. На студенческих командных соревнованиях участникам предлагается для решения от шести до двенадцати задач (вместо всего трех, как на школьных олимпиадах). Более того, уровень сложности большинства из этих задач обычно на порядок превосходит уровень задач для школьников. Часто бывает так, что идея решения задачи распадается на несколько частей, затрагивающих разные теоретические области, каждая из которых может быть довольно сложной в отдельности (как в плане идеи, так и в плане реализации). Да и сами условия предлагаются на английском языке (который хоть и знаком каждому, кто так или иначе связан с программированием, но все же является иностранным). От студента требуется уже более высокий уровень знаний и навыков отыскания решений и их дальнейшей реализации, чем от школьника. Однако в одиночку вникнуть в условия каждой из десятка задач (на иностранном языке), найти их решения, учесть все нюансы, предугадать возможные ошибки (а если этого не удалось сделать до этапа реализации, то суметь их отыскать и исправить) практически невозможно. Поэтому в командных соревнованиях необходимо распределять задачи между участниками команды, и кооперироваться при решении сложных задач.

Длительность тура. Соревнования студентов проходят в один тур, продолжительностью пять часов, без перерывов (в отличие от школьных, где бывает два тура по три часа, которые проводятся в разные дни). Пять часов беспрерывной напряженной работы, безусловно, требуют больших умственных усилий. Под конец тура у участников начинает сказываться усталость, новые идеи решений задач уже не так просто приходят в голову, искать ошибки становится сложней, внимательность при написании кода понижается и т. д. Тем более что обычно более простые задачи решаются на начальной стадии соревнований, а к концу тура остаются одни «гробы»[1].

Поэтому здесь очень важен опыт и регулярные тренировки. Постоянно участвуя в командных соревнованиях, вы со временем научитесь рассчитывать и чувствовать время тура, распределять силу так, чтобы пройти всю «дистанцию» в быстром, но ровном темпе. Однако, кроме самого по себе опыта, здесь еще важно правильно выбирать тактику и распределять свое время. Чтобы стало понятно, что я имею в виду, приведу такой пример.

Предположим, команда хорошо стартовала, быстро решила в начале простые задачи, хорошо шла в первой половине тура, но во второй в какой-то момент сложилась такая ситуация: двое участников совместно придумали решение сложной задачи, хорошо продумали все детали, вместе аккуратно ее реализовали, и сдали с первой попытки. В это время третий участник, решавший до этого другую задачу, наконец, после долгих попыток, обнаружил в ней небольшую ошибку, из-за которой она не проходила тесты жюри, исправил ее, и она также успешно сдалась (это вполне типичная ситуация). Казалось бы, все хорошо, команда успешно сдала две задачи подряд, все молодцы. Но если до этого никто из трех участников не думал над другими задачами, еще остающимися нерешенными (или что еще хуже: большинство членов команды еще даже не прочитали их условия), то возникнет большая пауза, в течение которой все участники будут вникать в новые условия задач. Только после этого у них начнут (возможно, с трудом) появляться какие-то мысли и идеи по поводу их решений, которые они будут сначала обдумывать, затем обсуждать друг с другом, продумывать детали реализации, и только потом писать программу за компьютером. В этом случае компьютер будет простаивать в течение длительного времени (это очень плохо, я еще подробнее поговорю об этом далее). Кроме того, даже если участники команды достаточно талантливы, и смогут придумать решения еще нескольких задач, сложится такая ситуация, что понятно, как писать две или более задачи, но компьютер только один, а тур медленно, но верно близится к своему концу…

И теперь возможны разные варианты: либо команда решит отбросить одну уже мысленно решенную задачу и программировать решение второй (и часто бывает непонятно, какой из двух задач в таком случае отдать предпочтение), либо разные участники станут в

[1] «Гробами» в сленге олимпиадного программирования принято называть очень сложные, неподъемные задачи.

спешке писать решения двух задач параллельно, по очереди уступая друг другу компьютер каждые 10 минут, для того чтобы подумать над своей задачей, и дать возможность своим «сокомандникам» писать решение другой. Второй вариант часто приводит к тому, что соревнование окончено, а у команды есть две недописанные (или даже уже написанные, но не до конца отлаженные) задачи, которые она так и не сдала. И все это лишь из-за неправильного распределения времени.

Один компьютер на троих человек. Хочу подчеркнуть важный нюанс соревнований формата *ICPC*: хоть в вашей команде и три человека, но компьютер-то на троих вам дается всего один! Что это означает? Грубо говоря, так как вас трое, то вы можете придумывать решения предложенных задач в три раза быстрее, чем, если бы каждый решал этот же набор задач поодиночке, но вот реализовывать эти решения на компьютере вы сможете лишь с прежней скоростью. Компьютер – вот узкое место команды. И этот критический ресурс должен использоваться максимально эффективно! В личных соревнованиях основным ресурсом является время. Если участник отвлечется на 10 минут, или неэффективно и безрезультатно потратит их на попытку придумать решение или отладить свою программу, то он потеряет в точности эти 10 минут (из всего трехчасового времени соревнований). В командных соревнованиях ситуация сложнее. Если один из участников потратит 10 минут времени, находясь за компьютером (например, продумывая сложные детали в процессе реализации алгоритма, или пытаясь найти ошибку в отладчике), то это время потеряет не только он (пусть даже добившись за эти 10 минут положительного результата), но и остальные члены команды, которые могли бы за эти десять минут начать писать другие задачи, решения которых им уже известны. Таким образом, будет потрачено в общей сложности 30 человекоминут.

В то же время, если участник тратит 10 минут на чтение условия, или на придумывание алгоритма, он никак не мешает своим «сокомандникам» в то же самое время параллельно использовать их собственное время на придумывание решений или написания реализаций других задач. Таким образом, в этом случае 10 минут на свои действия потратит только он сам. Если, скажем, двое из троих участников команды вместе обсуждают решение какой-то сложной задачи, то на 10 минут этого обсуждения они тратят в общей сложности 20 человекоминут из актива команды. Конечно, такие рассуждения немножко грубы, и в них в зависимости от ситуации могут проявляться различные нюансы, но, в общем и целом этот несложный, но очень значимый принцип вполне выполняется на практике, и его важно понимать и всегда иметь в виду при выборе своих очередных действий во время соревнования.

Критерии оценки правильности решений. В индивидуальных соревнованиях все решения задач тестируются и оцениваются один раз: только в конце тура. Результаты соревнований учитывают то, на сколько правильно были решены предложенные задачи. Решения прогоняются на заготовленном наборе тестов, и чем больше тестов проходит ваше решение – тем больше вы получаете очков, и, соответственно, тем выше ваше место в итоговой таблице результатов. При этом результаты таких олимпиад никак не учитывают ни то, насколько быстро вы решили ту или иную задачу (момент отправки вашего решения в жюри), ни то, решили ли вы ее правильно с первого раза, или неоднократно находили в уже сданном решении ошибки и исправляли их (а возможно даже, найдя концептуальную ошибку в идее решения, переписали все с нуля). Здесь важна лишь правильность итогового решения (последнего из тех, которые были посланы на проверку).

При этом подразумевается правильность в слабом смысле: решение может и не быть абсолютно верным, но если оно правильно работает для части случаев (а для остальных не работает из-за ошибки реализации или из-за неверного алгоритма), то вы все равно получите некоторую часть баллов. При этом, чем больше случаев ваше решение обрабатывает верно, тем больше баллов вам за него дадут.

Лично я, когда в школьные годы участвовал в различных личных олимпиадах и сборах, часто применял следующую тактику. В последние несколько минут соревнования, когда все

написанные решения уже проверены, а нормальных решений для оставшихся задач уже не написать, я писал на все нерешенные задачи «заглушки» («кирпичи», «затычки» и т. д.) – небольшие, короткие программы, которые решают данную задачу только на некоторых простейших вариантах входных данных, или с некоторой вероятностью угадывают ответ. Обычно такие «решения» можно придумать и написать довольно просто и быстро. Так я получал небольшие дополнительные баллы даже за те задачи, которые я не решил.

В командных же соревнованиях система проверки решений и оценки результатов совсем иная. Здесь команды сортируются в таблице результатов по числу решенных задач, а при одинаковом числе – по штрафному времени. При этом в качестве правильных зачитываются только те решения задач, которые проходят *все без исключения* тесты жюри. Заметьте, это сразу же говорит о том, что здесь никогда не имеет смысла писать частичное решение задачи, которое не будет работать на некоторых наборах входных данных. Более того, если вы придумали идею решения какой-то задачи, и вам она кажется правильной, но вы не уверены до конца, то перед тем, как бросаться ее реализовывать, лучше еще раз ее проанализируйте: действительно ли такое решение корректно обрабатывает все возможные варианты? В таком случае постарайтесь мысленно более или менее формально доказать, что это решение всегда будет выдавать правильный ответ. Подумайте, нет ли в вашем решении таких мест, в которых во время реализации могут всплыть различные многочисленные мелкие детали, требующие дополнительного обдумывания. Если таковые имеются, то их стоит детально продумать заранее. Удовлетворит ли полученная реализация вашего решения ограничениям по времени работы программы и по используемой памяти на *любых* тестах?

Если вы в чем-то сомневаетесь и сами не можете с уверенностью ответить на какие-то из вышеперечисленных вопросов, то лучше обсудите ваше решение с другим участником команды, прежде чем реализовывать его в одиночку. В противном случае вы рискуете, потратив свое время (а, главное, время использования компьютера), написать почти правильное или примерно правильное решение, которое, тем не менее, не принесет вашей команде никаких очков. Иногда, конечно, почти правильное решение после обнаружения и исправления ошибок можно превратить в действительно правильное, но бывает гораздо хуже, когда этого сделать нельзя. Тогда все ваши усилия окажутся, попросту, напрасными.

Принцип упорядочивания команд в таблице результатов. В личных соревнованиях команды упорядочиваются в таблице результатов в соответствии с общим числом баллов, набранных за решения задач. В командных соревнованиях все немного сложнее. Здесь команды сортируются по двум критериям: в первую очередь, по числу решенных задач, а при равенстве – по числу набранных очков штрафного времени (чем их меньше, тем выше место). Общее штрафное время команды исчисляется в минутах и вычисляется как сумма штрафного времени набранного по каждой успешно сданной задаче. Штрафное время по задаче – это время в минутах, прошедшее от начала тура до момента отправки правильного решения на проверку в жюри, плюс еще по двадцать дополнительных минут за каждую неправильную попытку (до первой удачной). Таким образом, для того чтобы минимизировать штрафное время, команда должна, во-первых, решать задачи как можно быстрее, а во-вторых – как можно реже посылать на проверку неправильные решения.

Сразу стоит обратить внимание на тот факт, что число решенных задач всегда важнее, чем штрафное время. К примеру, если команда *A* решила пять задач, а команда *B* – шесть, то последняя обязательно будет на более высоком месте, даже если она не сдала при этом ни одной задачи с первой попытки и ее штрафное время в пять раз больше, чем у *A*. Кроме того, так как штрафное время начисляется только за решенные задачи, всегда выгодно в конце тура пытаться сдать задачу любой ценой. Если вы при этом не сдадите задачу, то все ваши неудачные попытки ни на что не повлияют, ведь за них не будет начислено дополнительное штрафное время. Если же вы все-таки сдадите ее, то, хотя штрафное время в этом случае, возможно, сильно увеличится, но занимаемое вашей командой место может только улучшиться, так как теперь вы уже решили на одну задачу больше.

Однако не стоит посылать множество неправильных решений в середине тура, так как здесь ситуация уже иная. У вас еще есть время, чтобы аккуратно найти и исправить возможные ошибки самим, пусть даже потратив чуть больше времени, вместо того, чтобы наобум вносить случайные изменения, пытаясь угадать правильное решение. Если в обоих случаях вы, скорее всего, и сдадите эту задачу, то в первом вы при этом заработаете куда больше лишнего штрафного времени. А избавиться от уже сделанных неудачных попыток и уменьшить свое штрафное время уже никак не возможно.

Этот пункт в сочетании с предыдущим позволяет сделать весьма важный и интересный вывод. Условно его можно сформулировать так: в личных соревнованиях важнее хорошо протестировать свои уже написанные решения на предмет наличия ошибок реализации перед их проверкой жюри (перед концом тура), в то время как в командных – важнее убедиться в правильности идеи своего алгоритма и в том, что он учитывает все возможные случаи, а также всегда удовлетворяет ограничениям на используемые ресурсы (время и память). В личных соревнованиях нет возможности послать свое решение на проверку еще раз, после того, как оно не пройдет почти все тесты жюри, кроме пары-тройки, из-за того, что вы, скажем, в одном месте по ошибке написали «`i`» вместо «`j`». В то же время в командных соревнованиях вы не получите никакой пользы от того, что напишете решение, которое на 90% тестов выдаст верный ответ, или, к примеру, решение, которое всегда работает правильно, однако на самых больших допустимых входных данных работает на пол секунды дольше, чем того требуют ограничения, установленные жюри. Этот принцип стоит иметь ввиду, особенно если вы уже хорошо себя проявили на личных соревнованиях, но только собираетесь участвовать в командных.

Сразу оговорюсь, что это вовсе не означает, что в личных соревнованиях не стоит пытаться придумывать правильный алгоритм вместо пусть даже хорошей эвристики. Это также не значит, что, участвуя в командных турнирах, не стоит вообще тестировать свое решение перед отправкой в жюри. Конечно, иногда можно ограничиться проверкой только тестов из условия, если задача очень уж простая, или тур подходит к концу. Или, если вы до этого уже как следует протестировали все возможные случаи, а теперь нашли и исправили специфическую ошибку, которая не влияет на суть алгоритма (ошибка в структуре данных, частный случай типа `n = 0` и т. д.), можно проверить лишь то, что решение работает на том хитроумном тесте, на котором оно раньше не работало (и опять же на тестах из условия). Однако почти всегда лучше все же тщательно проводить тестирование. В конце концов, я за свою жизнь не встречал ни одной команды, для которой проблемой было то, что ее участники тратят на тестирование слишком много времени.

Интерактивность тестирования. Другим важным отличием личных соревнований от командных является то, что в личных соревнованиях никто из участников не имеет никакой информации о правильности или неправильности своих решений и о том, какие задачи решили (или хотя бы пытались решить) другие участники вплоть до окончания тура. В командных соревнованиях такая информация доступна участникам во время тура.

В соревнованиях формата *ACM ICPC* каждое решение тестируется сразу же, как только команда отправляет его на проверку в жюри, а результат тестирования сообщается команде. Проверка производится следующим образом: решению на вход по очереди подаются заготовленные жюри тестовые наборы входных данных (тесты) для данной задачи. Каждое решение запускается на каждом таком наборе, и соответствующие выходные данные, которые оно выдает, проверяются на корректность. Кроме того, каждый раз проверяется, что выполнение программы не вызвало ошибку, и что оно уложилось в установленные ограничения по объему используемой памяти и по времени выполнения. Такая проверка производится до тех пор, пока не встретится первый тест, который решение не проходит, или пока оно успешно не пройдет все тесты жюри. В первом случае номер теста, на котором решение «свалилось», сообщается команде вместе с типом ошибки, а во втором – решение засчитывается как правильное.

Напомню все возможные типы сообщений, которые может получить команда в результате отправки решения на проверку (впрочем, они стандартны и используются как в командных, так и в личных соревнованиях):

- **Accepted (OK)** – Программа успешно прошла все тесты.
- **Compilation error (CE)** – Ошибка компиляции. В программе содержится синтаксическая ошибка, и компилятор не может ее корректно скомпилировать. Эта ошибка возникает редко, и обычно бывает вызвана невнимательностью участников. Чаще всего она возникает, когда при посылке решения было по ошибке указано не то имя файла или не тот тип компилятора.
- **Time limit exceeded (TL)** – Превышение предела времени. Программа на данном тесте работала дольше допустимого ограничения по времени (обычно 2 с), и была прервана тестирующей системой.
- **Memory limit exceeded (ML)** – Программа в какой-то момент времени при решении данного теста превысила допустимый объем используемой памяти.
- **Runtime error (RE)** – Ошибка времени исполнения. В процессе выполнения программа сгенерировала ошибку и аварийно завершилась (например, индекс вышел за границы массива, была попытка обработать нулевой указатель, произошло деление на ноль и т. д.). Этот же результат будет получен, если код возврата при выходе из программы принимает значение, отличное от нуля.
- **Security violation (SV)** – Нарушение ограничений безопасности. Этот тип ошибки очень похож на предыдущий. Он означает, что программа попыталась совершить какое-то запрещенное действие, например, обратиться в чужую область памяти или попытаться создать файл вне текущей директории.
- **Wrong answer (WA)** – Неправильный ответ. Решение отработало без ошибок, но ответ, который был получен в результате, не является верным.
- **Presentation error (PE)** – Неправильный формат выходных данных. Похож на предыдущий тип, за исключением того, что здесь уже сам формат выходных данных является неверным. Например, требовалось вывести целое число, а программа вывела число с плавающей точкой, несколько чисел, произвольную строчку, создала пустой файл или вывела ответ не в тот файл.

Все сообщения, кроме первых двух, содержат номер первого теста, на котором обнаружилась ошибка. Замечу также, что на практике в силу специфики проверяющей программы классы ошибок могут часто не соблюдаться строго, и, например, вместо ошибки **PE** тестирующая система часто может выдавать ошибку **WA**, а ошибка **ML** в разных системах иногда может выдаваться как **RE**.

Тесты по конкретной задаче всегда прогоняются в одном и том же заранее установленном порядке, одинаковом для всех. Обычно хороший набор тестов по одной задаче содержит примерно 20-40 тестов, однако это не строгое правило. Почти на всех соревнованиях первыми тестами из набора жюри по данной задаче являются тесты из ее условия, указанные в качестве примеров (об этом обычно говорится в правилах соревнований).

Из сообщений об ошибках, полученных после проверки решения, участники команды могут получить много ценной информации. Рассмотрим несколько примеров:

*Предположим, вы послали решение на проверку, и получили ответ **PE 1** (цифра обозначает номер теста). Это означает, что вы невнимательно протестировали ваше решение на тестах из условия, и не заметили ошибку в формате вывода (лишний пробел в середине, "IMPOSSBILE" вместо "IMPOSSIBLE" и т. д.), либо неправильно назвали имя выходного файла. Исправьте ошибку и пошлите решение еще раз. Пусть теперь вы получили ответ **WA 5**. Это значит, что ваше решение уже на пятом тесте выдает неправильный ответ. Чаще всего (но не всегда) в этом случае ошибка довольно простая, и ее можно быстро найти, читая условие, или проверив несколько простых тестов на компьютере. Предположим, вы исправили мелкие ошибки, и после очередной отправки на*

*проверку ваше решение получает ответ **TL 35**. В этом случае, ваш алгоритм с большой вероятностью всегда выдает правильный ответ (так как на тестах с первого до 34 он сработал верно), однако, либо вашей программе требуется некоторая оптимизация, чтобы она работала немного быстрее, либо, что хуже, жюри ожидало для этой задачи асимптотически более быстрое решение. Если вы теперь произвели некоторые улучшения, и, отправив решение на проверку, получили ответ **WA 27**, то это означает, что вы в процессе оптимизации внесли в программу ошибки. В этом случае, стоит внимательно перечитать именно те участки кода, которые вы добавили или изменили перед отправкой решения. Если бы в предыдущем случае вы получили, к примеру, **WA 48**, то уже нельзя было бы утверждать, что ошибка внесена при последнем изменении (так как предыдущие версии программы вообще не тестировались на тесте 48). Кроме того, в этом случае становится понятно, что ошибка довольно специфична и сложна, и тестированием на простых или случайных тестах ее уже едва ли можно будет обнаружить. Скорее всего, это либо некоторый каверзный частный случай, который вы упустили из виду в процессе решения, либо трудно заметная опечатка в программе, которая влияет на правильность ответа только в редких особых случаях.*

Наличие монитора. Другим источником информации о ходе соревнования, доступным участникам командных турниров во время тура, является интерактивный монитор. Монитор – это таблица результатов, которая показывает позиции всех команд на текущий момент времени, а также множество дополнительной полезной информации. Каждой команде соответствует строка монитора, в которой помимо ее текущего места указано то, сколько всего задач решила эта команда и какое у нее сейчас суммарное штрафное время. Кроме того, для каждой из предложенных задач показано, сдала ли команда эту задачу, и, если сдала, то на какой минуте соревнований и с какой попытки, а в противном случае – сколько неудачных подходов по этой задаче уже было предпринято данной командой. В первом случае в соответствующей ячейке монитора будет запись вида «+X», где X – число неудачных попыток, предпринятых командой по этой задаче (общее количество посланных решений минус один). Например, если команда сдала задачу с третьей попытки, то в ячейке монитора будет записано «+2». Вместо «+0» обычно пишется просто «+». Во втором случае в соответствующей ячейке будет запись вида «−X», где X – число неудачных подходов к задаче. В обоих случаях в ячейке вместе с числом попыток по данной задаче обычно отмечается время отправки последнего решения на проверку.

Монитор доступен как тренерам и зрителям, так и самим участником команд на протяжении тура, однако за час до окончания соревнований его результаты перестают обновляться (в этом случае говорят, что монитор «заморожен»). Это делается для того, чтобы не выявлять победителя соревнований сразу после окончания тура, а сохранить интригу до церемонии награждения.

Какая информация, предоставленная в мониторе, должна быть наиболее интересна участникам команды? Казалось бы, ответ очевиден – место этой команды в таблице результатов. Однако, это отнюдь не так. Конечно, все хотят выступить как можно лучше, всех волнует результат. Тренеры и руководители команд на протяжении всего тура болеют за своих воспитанников, не сводя глаз с монитора. Каждая успешно сданная задача вызывает у них бурю ликования, каждая неудачная попытка или длительная пауза – сильное волнение и огорчение. Сидя в тренерском зале, они суммируют штрафное время команд, пытаются просчитать все возможные варианты развития хода соревнований. *Что будет, если команда A сдаст эту задачу в течение 30 минут с первой попытки? А если не с первой? Обойдет ли она команду B с таким штрафным временем? У команды C уже три неудачные попытки по задаче. Значит ли это, что она безнадежно увязла, или, напротив, скоро найдет и исправит ошибку?*

Все эти вопросы во время тура должны волновать тренеров, но вовсе не участников команд. Последние должны быть по возможности хладнокровны. Ведь перед ними стоит одна цель, которая не зависит от текущего положения в соревновании: выступить как можно

лучше, то есть решить как можно больше задач, получив при этом как можно меньше штрафных очков. При выполнении этой цели занятое в итоге место автоматически будет максимально высоким. И не важно, на каком месте ваша команда сейчас находится: все равно имеет смысл пытаться решать задачи как можно лучше. Есть мнение, что во время тура основными противниками команды являются не другие команды, а предложенные задачи, и я с ним полностью согласен. Уже после окончания тура можно попытаться узнать, сколько задач решили другие команды за последний час, оценить свои шансы занять то или иное место в итоговой таблице, подумать: «А что было бы, если бы мы все-таки решили эту задачу на 10 минут раньше и без ошибки?», и т. д. Во время тура на это не следует отвлекаться. Поэтому ситуация, когда все трое участников команды минутами смотрят в монитор, изучают свою позицию и позиции своих соперников и оценивают свои шансы занять то, или иное место, вместо того, чтоб решать задачи, недопустима.

Означает ли это, что командам лучше вообще не открывать монитор? Вовсе нет. В монитор стоит периодически заглядывать, просто следует концентрироваться на другой информации, а именно, на информации по той или иной задаче. В начале тура желательно смотреть в монитор для того, чтобы найти «халяву» – задачу, которую можно просто и быстро решить. Именно в ее столбце таблицы монитора появятся первые плюсы. К середине тура по монитору будет видна примерная картина сложности задач. К примеру, если по какой-то задаче есть целый столбец плюсов, а вы над ней еще толком не думали, то стоит заняться ей: наверняка она решается несложно. Если же вы придумали решение какой-то задачи и собираетесь его запрограммировать, однако в мониторе по этой задаче стоят в основном минусы или плюсы с большим числом попыток, то стоит задуматься: точно ли вы ничего не упустили в своем решении? Может быть, оно не обрабатывает какие-то частные крайние случаи (чего не замечали ваши соперники, когда посылали свои решения на проверку и получали свои минусы)? Или, быть может, задача такова, что в процессе реализации возникнет множество мест, где можно из-за невнимательности совершить ошибку?

Еще часто бывает так, что у некоторой задачи существует не слишком сложное решение с некоторым асимптотическим временем работы (к примеру, $O(n^2 \log n)$), а ограничения на размер входных данных задачи таковы, что не ясно, подразумевалось ли авторами именно это решение, и пройдет ли оно по времени, или требуется придумать что-либо более хитроумное (скажем, с асимптотикой $O(n^2)$). В таком случае, столбец из множества плюсов будет означать, что, скорее всего, простое решение проходит и все его успешно сдают. Ну а если эту задачу сдает мало команд, и далеко не с первой попытки, или по ней вообще почти одни минусы, то, скорее всего, такое решение будет работать слишком долго, и не зачтется, либо придется его долго и щепетильно оптимизировать перед тем, как все-таки «протолкнуть»[2].

Таким образом, команды-соперники на время тура действительно становятся союзниками, волей-неволей подсказывая друг другу, за какие задачи стоит браться, а за какие нет.

Когда же стоит заглядывать в монитор? Лучше всего это делать в момент отправки очередного решения на проверку в жюри. В этот момент участник, писавший решение, как раз открывает программу посылки решений на проверку (которая обычно проинтегрирована с монитором). Если открывать монитор во время того, как один из участников программирует решение за компьютером (а так должно быть почти всегда на протяжении тура, так как компьютер не должен простаивать), то ему придется отвлекаться, и это может привести к тому, что он допустит в решении какую-нибудь ошибку.

При этом не стоит долго изучать монитор: пары десяток секунд вполне хватит на то, чтобы оценить обстановку, и решить, на какие задачи стоит переключить свое внимание.

[2] На сленге «протолкнуть задачу» – это значит сдать ее, но с трудом, не с первой попытки, или с не совсем правильным или медленным решением (после оптимизаций).

Общие советы

Далее я приведу несколько общих соображений по поводу тактики участия в командных турнирах.

С точки зрения стратегии и тактики, пятичасовой тур командных соревнований, подобно шахматной партии, принято условно делить на начало, середину и конец. В первые минуты тура очень важно не терять времени, и сразу же начать в быстром темпе читать и решать задачи. Участникам обычно предлагаются задачи разного уровня, и практически в каждом турнире среди них есть одна – две довольно простые («халява»). Так как при подсчете результатов тура сложность задач никак не учитывается, а вот время решения, напротив, играет роль, то очень важно сразу же найти и решить все простые задачи.

Имеет смысл сразу же после начала тура написать общий шаблон программы решения. Во всех программах-решениях, кроме, собственно, реализации алгоритма, выполняются некоторые подготовительные действия: подключение необходимых стандартных библиотек, открытие и закрытие файлов, названия которых имеют определенный формат. Если формат соревнований подразумевает, что все тесты подаются на вход программы в единственном файле, то в каждом решении будет присутствовать общий цикл, который последовательно читает файл и вызывает процедуру, которая считывает данные одного теста и вычисляет его ответ. Все эту рутину и следует вынести в шаблон. Его написание займет у вас минуту, однако так как вы будете использовать этот шаблон в каждой задаче, то его написание сэкономит вам как минимум десяток минут компьютерного времени, которые могут пригодиться в конце тура.

Формат шаблона выработать несложно, но важно сделать это сознательно, с учетом особенностей стиля программирования в вашей команде. Затем следует выбрать из вашей команды одного человека, который будет писать шаблон в начале каждого тура (а также, возможно, настраивать компилятор). Через несколько тренировок этот человек научится делать все это не задумываясь, почти мгновенно.

Остальные участники сразу же после начала тура должны читать задачи и искать среди них простые, причем желательно найти их как можно быстрее. Наша команда в свое время действовала так: один участник сразу же после начала тура садился набирать шаблон решения, в это время остальные двое читали условия задач, причем один начиная с первой, а второй – начиная с последней. После того, как первый заканчивал набирать шаблон, он тоже переключался на чтение задач, причем начинал с середины. Благодаря этому уже после нескольких минут от начала старта у нас был уже написан шаблон и найдена простая задача (так как все задачи были прочитаны хотя бы одним человеком). Сразу же после обнаружения «халявы» тот, кто ее нашел, садился ее писать, а остальные участники дочитывали условия остальных задач.

Важно в начале тура не кидаться писать первую же прочитанную задачу. Часто бывает так, что из условия задачи сразу же понятно, что в ней требуется сделать и как ее решать, однако это вовсе не всегда означает, что задача на самом деле простая. Есть такой класс задач, которые в программистско-олимпиадном сленге называются «техническими гробами». Эти задачи бывают простыми с идейной стороны и часто предполагают применение различных стандартных алгоритмов для своего решения. Однако эти алгоритмы могут быть сложными с точки зрения реализации. Типичная разновидность такого вида задач обычно выглядит так: условие длинной в несколько страниц, в котором описывается некоторая замысловатая система или процесс со множеством деталей, условий и пунктов, и как бы говорится почти в явном виде: «напишите то, что здесь описано». Опытные участники сразу же определяют такую задачу. Даже если все составные алгоритмы, которые требуются для ее решения, довольно просты, в сумме решение подобной задачи обычно получается очень громоздким и занимающим много времени для написания. Кроме того, при реализации такого решения велика вероятность допустить ошибку.

Соблазн сразу же начать писать, пусть даже очень сложное решение, суть которого понятна сразу, очень велик. И все же подобные задачи стоит решать, только если у вас совсем нет идей по остальным задачам. И уж никак нельзя решать их в начале тура.

Сразу стоит сказать, что если уж вы собираетесь писать решение технической задачи, то, продумайте как следует все детали его реализации, убедитесь, что в нем нет непонятных и слишком сложных мест, при написании которых придется надолго задуматься. Если же они есть, то аккуратно все продумайте заранее, выпишите все сомнительные участки решения на отдельный лист бумаги для того, чтобы все учесть и ничего не забыть.

Вообще написание части кода решения трудной задачи заранее на бумаге – это применимая, и очень даже неплохая тактика. На первый взгляд это кажется абсурдом: зачем тратить время и писать что-то два раза? Тем более что писать код за компьютером, в среде разработки на порядок проще и удобнее, чем на бумаге. Однако в действительности это позволит вам сэкономить драгоценное компьютерное время. Просто начните писать основную часть алгоритма решения сложной задачи так же, как вы бы делали это на компьютере, и сразу же, как вы начнете это делать, начнут всплывать всякие нюансы и подробности, которые раньше не были видны. Вы с первых же строчек поймете, что что-то не учли, вам придется что-то зачеркнуть и переписать по-другому. В итоге вы будете видеть детали своего решения гораздо лучше. Сразу же всплывут многие подводные камни. Часто достаточно просто аккуратно выписать все переменные, которые будут использованы в вашем алгоритме, для того чтобы сразу верно оценить его глубину и возможные проблемы. Важно, что в то время, пока вы пишете и продумываете код на бумажке, компьютер эффективно используется кем-то другим из вашей команды. При этом, когда вы сядете за компьютер, то потратите на порядок меньше времени, так как не будете останавливаться и мучительно задумываться над тем, как стоит написать ту или иную часть алгоритма – вы просто будете переписывать код с бумажки. Очень сложно приучить себя использовать этот прием, однако иногда он бывает очень полезен.

Важно, чтобы в начале тура ВСЕ участники прочитали ВСЕ предложенные задачи. Один человек не всегда в состоянии точно оценить сложность задачи, и вероятность придумать ее решение. В то время как одному участнику задача может показаться слишком сложной, и он отложит ее, даже толком не подумав над ней, другому может прийти оригинальная мысль, позволяющая придумать довольно простое решение. Кроме того, один человек может ошибиться, пропустить в условии задачи важное дополнительное условие, неправильно интерпретировать какую-то фразу.

По этой же причине никогда не стоит рассказывать условия задач друг другу. Если вы хотите обсудить идею решения какой-то задачи со своим партнером по команде, а он его еще не читал (а во второй половине тура это уже само по себе осень плохо), то вместо того, чтобы объяснять ее суть, дайте ему время самому внимательно прочесть условие.

Еще один небольшой совет – во всю используйте возможность печати решений для поиска ошибок: всякий раз, когда вы посылаете решение на проверку – посылайте его также и на печать, не дожидаясь ответа тестирующей системы. На серьезных соревнованиях количество бумаги, доступное одной команде для печати, достаточно велико или вовсе не ограничено, так что вы ничего не будете терять, а в случае, если посланное решение все же ошибочно, сэкономите себе время.

Для поиска ошибок в решении стоит пользоваться именно распечатками, а не пошаговым выполнением (*debug*). Это ошибочное мнение, что при помощи пошагового выполнения программ можно легко найти ошибку. Пошаговое выполнение – кропотливый процесс, и занимает много времени, а если логика алгоритма довольно сложна, он состоит из множества частей, и непонятно, в которой из них ошибка, то уловить место ее первого появления будет очень и очень сложно. Кроме того, все время, пока вы будете отлаживать программу, компьютер будет занят. Если же чтение распечаток все же ничего не дает, то хорошей альтернативой пошаговому выполнению может стать вывод промежуточных данных на консоль прямо в процессе работы алгоритма (так называемый *debug output*).

Главным его преимуществом является то, что при этом вы сразу же можете видеть все промежуточные состояния тестируемой программы, а не только состояние в текущий момент времени. Кроме того, добавив один раз вывод промежуточной информации, вы сможете использовать его многократно. При этом следует не забывать его комментировать всякий раз перед отправкой решения на проверку, если программа должна выводить основной ответ задачи на консоль, или если алгоритм критичен по времени (так как вывод большого объема информации может существенно тормозить работу программы).

Заключение

Я попытался рассмотреть главные особенности командных соревнований и сформулировать некоторые советы по стратегии и тактике участия в них. Большинство из этих советов в свое время я получил от Андрея Сергеевича Станкевича, выдающегося олимпиадного программиста и тренера, который является самым молодым лауреатом премии Президента РФ в области образования. Именно благодаря его огромным усилиям и умелому руководству нашей команде, как и ряду других команд нашего университета, удалось достичь столь высоких результатов на олимпиадах по программированию.

Многие аспекты олимпиадного программирования не были рассмотрены в этой статье или были описаны лишь обзорно, однако я надеюсь, что приведенные советы и идеи помогут вам успешно выступать в командных соревнованиях разного уровня.

Желаю вам удачи и многих побед, как в олимпиадном программировании, так и в жизни!

Список литературы

1. *Оршанский С.* О решении олимпиадных задач по программированию формата ACM ICPC //Газета для учителей «Информатика». 2006. № 1. http://inf.1september.ru/article.php?ID=200600106

2. *Миронов И.* Теория. Как принять участие в олимпиаде и получить от этого удовольствие. Практика. Тернистый путь к успеху. Командный чемпионат мира по программированию ACM 2007/2008. Северо-Восточный Европейский регион. СПб.: СПб ГУ ИТМО, 2007.

3. *Волков Л., Шамгунов Н.* Как стать чемпионом Урала по программированию. Командный чемпионат мира по программированию ACM 2007/2008. Северо-Восточный Европейский регион. СПб.: СПб ГУ ИТМО, 2007.