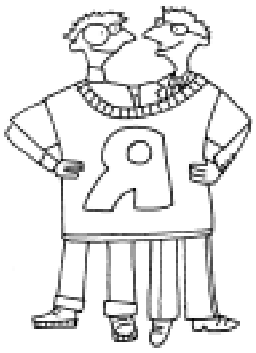
*Шамгунов Никита Назимович,*
*Волков Леонид Михайлович*

# КАК СТАТЬ ЧЕМПИОНОМ УРАЛА ПО ПРОГРАММИРОВАНИЮ

Когда осенью 1996 года две команды УрГУ вернулись с первого питерского полуфинала, заняв там 8 и 9 места, члены этих команд, возможно, чувствовали себя бесконечно умудренными опытом людьми. По горячим следам они (и в их числе первый из авторов этой статьи) написали замечательную книжку «Как стать чемпионом мира по программированию». Осмелимся предположить, что эта книжка в свое время многих подвигла на активные занятия АСМ-олимпиадами, как подвигла она второго из авторов этой статьи. Тогда казалось, что накопленный опыт почти достаточен, чтобы выиграть этот самый чемпионат мира...



*...наше коллективное «я»...*

Прошли годы. Сменились поколения. В УрГУ, как и в других вузах, уже совсем другие люди играют в АСМ-программирование. Наши взгляды на спортивное программирование прошли шлифовку временем и бесконечными баталиями. Максимализм уступил место зрелому опыту. Не будем писать о том, в чем мы не можем считать себя экспертами, – оставим это Андрею Лопатину и Николаю Дурову*. Лучше напишем о том, что каждому из нас удавалось трижды: как стать чемпионом Урала по программированию.

Мы теперь, увы, работаем в разных городах, и статью пришлось собирать из разных фрагментов, написанных от первого лица. При окончательном сведении текста каждое «я» мы сначала хотели заменить на «я» (Н.Ш.) или «я» (Л.В.). Но за годы игры в одной команде у нас сложился достаточно общий взгляд на те вещи, о которых пойдет речь ниже, и поэтому мы решили оставить все «я» в тексте без изменений. Будем считать, что это наше коллективное «я», эдакое распределенное Санкт-Екатеринбургское «я» рассказывает о своем опыте выступлений в первых пяти чемпионатах Урала по программированию.

## ЧТО ТАКОЕ – «БЫТЬ ЛУЧШЕ»

Правилами совершенно четко определены условия победы на олимпиаде по программированию по регламенту АСМ. Команда должна решить больше задач, чем другие команды, а при равенстве количества решенных задач – заработать меньше штрафного времени. Итак, для того, чтобы выигрывать на олимпиаде по программированию, нужно быть лучше других. И это самое «лучше» состоит из многих факторов. Для победы в соревновании на комплекте весьма различных задач необходимо найти решение по возможности большего количества задач и качественно, чисто, быстро запрограммировать каждую из решенных.

## КАК РЕШАТЬ ЗАДАЧИ

Я не знаю, как решать задачи. Я знаю только, что после того как решишь их много, начинаешь делать это лучше, начинаешь лучше видеть возможные подходы к решению задач, начинаешь лучше их чувствовать. И все равно не последнюю роль

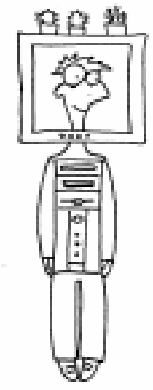
* Члены команды СПбГУ – чемпиона мира 2000 и 2001 года, в настоящий момент Андрей Лопатин – тренер команды СПбГУ. *Прим. ред.*

здесь играет фактор «способностей», «таланта»... Того, что словами не определить и никаким аршином не измерить. Фактор неопределенности, который в итоге ранжирует абсолютно равные по подготовке команды. Поэтому в дальнейшем мы будем считать, что все команды, выходящие на старт очередного контеста*, абсолютно равны по своим способностям. Постараемся показать теперь, как можно максимизировать остальные параметры, определяющие успех команды. Ведь решить задачу – это полдела, необходимо еще и запрограммировать решение.


## МАШИНА ДЛЯ УКЛАДКИ КИРПИЧЕЙ

Реализация – это в некотором смысле не такое творческое дело, как собственно решение задачи. Говоря так, мы нисколько не хотим обидеть талантливых кодеров, которые истинное наслаждение находят в виртуозном использовании ресурсов языка для оптимальной реализации казалось бы банальных вещей. Мы лишь хотим подчеркнуть, что умение качественно реализовывать решение в гораздо большей степени поддается тренировке и улучшению в сравнении с умением решать задачи.

Единственный путь победить в качественной реализации – это воспитать в себе машину, которая при программировании очень редко обращается к голове. Голова при кодировании может подключаться к процессу лишь в критические моменты и обязательно не в тот момент, когда кодировщик находится за компьютером. В основном же кодировщик должен уметь собирать программу из заготовленных кирпичиков. Чем больше под рукой готовых шаблонов, тем быстрее и надежнее соберется программа. Если вам придется придумывать, как сделать тот или иной блок, человек, у которого он уже есть – сидит в пальцах, как говорится, – несомненно обгонит вас в реализации.


...воспитать в себе машину...

Здесь на помощь может прийти такая практическая рекомендация: собирают задачу из кирпичиков до того, как сядут за машину. Нормой должно стать следующее правило: все задачи, начиная со второй, должны кодироваться путем перепечатывания с бумажки. Таким образом, получаем, что первой задачей должна быть некоторая технически очевидная задача – при этом она необязательно должна быть самой легкой.

Сила подхода, который основывается на программировании из готовых шаблонов, заключается в том, что реализация готовых алгоритмов доводится до автоматизма, и в этом случае большие и сложные участки кода становятся сразу отлаженными. Тогда отлаживать необходимо лишь соединительные участки кода – а они почти всегда меньше и, к тому же, логически проще.

Что такое эти строительные блоки? Это стандартные алгоритмы: длинная арифметика, построение НОК и НОД, графовые алгоритмы и многое другое. Но не только они. Минизадачи по вводу данных, по представлению данных в памяти, по реализации длинных массивов также относятся к кирпичам. Здесь же – и классы для работы с геометрией, приемы построения конечных автоматов для задач на симуляцию и многое другое. Хорошие реализации на дороге не валяются. Почти наверняка то, что вы при-

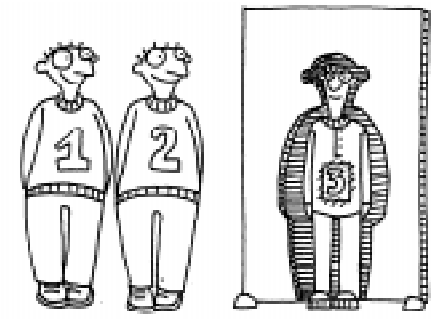
* Авторы используют сленг, видимо, уже возникший среди участников чемпионата мира по программированию: *contest* – соревнование (англ.) *Прим. ред.*

думаете и реализуете в первый раз, хоть и будет работать, тем не менее не будет оптимальным по красоте, по ясности кода и, вполне возможно, по эффективности. Необходимо тщательно анализировать и непрерывно совершенствовать свою коллекцию заготовок. Есть легкий способ определить, хорошо ли написана та или иная программа: если она написана просто, изящно и все в ней понятно, значит, вы на верном пути. Для достижения этой цели нужно читать книги, решать задачи и анализировать чужой код. Причем, если в книге излагается уже знакомый вам алгоритм, не стоит его пропускать: возможно какие-то новые мысли придут вам в голову при его чтении. Хороший пример – книга Кормена[*]. Я пересмотрел свои представления о построении потока в сети, о поиске подстроки в строке, научился некоторым новым алгоритмам и узнал несколько новых способов сортировки.

### КЛОН САМОГО СЕБЯ

Для формирования более ясного представления о том, что же такое хорошо, а что такое плохо, нужно прорешать множество задач на judge и timus[**]. Желательно списываться с другими участниками и просить их прислать решения тех задач, код которых вас не удовлетворяет. Еще более полезно скачивать решения жюри полуфинальных соревнований. Особенно хорошие задачи традиционно в Санкт-Петербурге, Чехословакии, Канаде и Германии. После прорешивания этих задач сравните ваши решения с решениями жюри. Если ваши решения лучше (короче, изящней, понятней) – поздравляю. Если нет – поймите, чем ваши решения хуже, и переучитесь. Анализ кода – это не копание в грязном белье, а очень достойное занятие, которое может многому научить. Постарайтесь писать одинаковые вещи АБСОЛЮТНО одинаково. Это необходимо, чтобы не ошибаться в простых случаях. Тем не менее, если находите лучший способ реализации – переучивайтесь. Я лично почерпнул от канадцев много идей.

*Постарайтесь писать одинаковые вещи АБСОЛЮТНО одинаково.*



*...все должно быть унифицировано.*

Если у вас уже что-то сидит в пальцах, – не забывайте регулярно это освежать и повторять. Возвращайтесь снова к задачам, которые вы решали когда-то давно. Если когда-нибудь вы сможете написать решение, повторяющее прошлогоднее байт в байт, – это повод для радости.

### ТРИ КЛОНА

Следующий этап, к которому необходимо стремиться, – это стандартизация представлений внутри команды. Отступы, правила использования функций, ввод и вывод, именование переменных и функций – все должно быть унифицировано. Тогда и перекрестная отладка будет быстрее, ведь меньше надо друг другу объяс-

---

[*] *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построения и анализ. М.: МЦНМО, 2000.

[**] Сервера с тестирующими системами. В настоящий момент существует, по крайней мере, четыре таких сервера: acm.timus.ru – сервер МГУ, acm.sgu.ru – сервер Саратовского государственного университета, acm.uva.es – старейший сервер такого рода (Испания), acm.zju.edu.cn – китайский сервер. *Прим. ред.*

нять. Очень полезно брать программу и объяснять другому человеку, как она работает. Общий стиль – это то, к чему надо стремиться постоянно.

Я очень хорошо помню момент, когда мы «параллелили» в Петербурге задачу про домино и, не сговариваясь, одинаково назвали функцию, по которой стыковались наши модули. Именно в этот момент, за 25 минут до конца контеста, я поверил, что мы будем в финале.

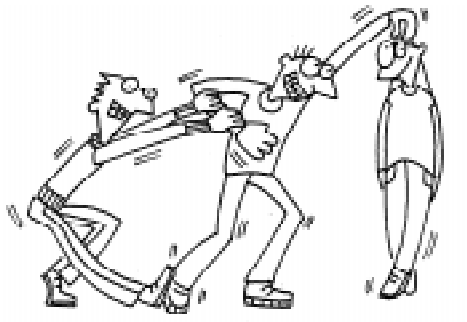
### ТЕХМИНИМУМ

Что касается того, какие конкретно алгоритмы надо знать: необходимо знать все! Правда, лучше изучать их в порядке возрастания сложности. Что толку от того, что вы знаете, как программировать венгерский алгоритм, если вы не умеете раскладывать число на простые множители. Вторая задача встретится в сто раз чаще. Очень полезно в этом плане решать задачи из книги Матюхина, Пономарева*.

Однако же, нельзя объять необъятное. И поэтому достаточно давно в нашей команде была выработана практика, которую я считаю очень полезной. А именно, очерчен некоторый круг задач, который мы посчитали техминимумом, обязательным для каждого. Остальные типы задач были поделены между участниками команды для специального изучения. В техминимум были включены базовые, общеизвестные алгоритмы и, что немаловажно, кирпичики, которые неизбежно возникают при программировании алгоритмов любой сложности.

### ТРЕНИРОВКИ

Старайтесь разнообразить ваши тренировки. Не ограничивайтесь моделированием типового контеста. Достаточно легко придумать разные виды тренировок, личных, командных и лично-командных, которые помогают оттачивать те или иные аспекты мастерства.



*...работайте над моральным климатом в команде.*

Вот, например, после рабочего дня редко остаются силы на полноценную тренировку. Да и смысла в такой тренировке будет немного. В этой ситуации полезно взять 2–3 трудных задачи на 1,5–2 часа, причем это могут быть задачи, над которыми раньше вы уже думали. Таким образом отлично моделируется ситуация финиша (и рабочий день позади добавляет естественности), который так часто является камнем преткновения для многих команд.

Отличной тренировкой является придумывание задач, подготовка соревнований школьников, придумывание тестов к задачам. Наконец, полезно решать исследовательские задачи, требующие погружения в специальную литературу и нескольких дней размышлений.

### ПОЛУЧАЙТЕ УДОВОЛЬСТВИЕ

Очень важен и психологический аспект. Я помню множество примеров, когда сильная команда «перегорала» в ответственных соревнованиях из-за психологического груза на ее плечах. Замечательно, когда команда представляет собой коллектив единомышленников. Если нет – работайте над моральным климатом в команде. Не очень хорошо, когда в команде есть ярко выраженный лидер, на котором лежит ответственность за все решения; гораздо лучше – но труднее – выпестовать командное «я», которое умеет принимать правильные решения даже в самых трудных ситуациях.

---

* *Беров В.*, *Лапунов А.*, *Матюхин В.*, *Пономарев А.* Особенности национальных задач по информатике. Киров: ООО Триада-С, 2000.

Старайтесь получать удовольствие от самого процесса игры – это очень важно. Будьте азартны, будьте уверены в себе. Если вы научились получать удовольствие от игры, то не следует тренироваться в последние дни перед важным соревнованием: тогда вы не будете испытывать пресыщения, а сможете рваться в бой.

**РЕЗЮМЕ**

1. Программируйте.

Следуйте принципу: «Если хочешь научиться программировать – программируй». Постоянно смотрите по сторонам, как люди программируют, это не даст вам сбиться с дороги.

2. Изучайте алгоритмы.

В них заключено огромное количество идей. Доводите до автоматизма свои реализации этих алгоритмов.

3. Унифицируйте.

Добивайтесь в своей команде одинакового взгляда на качественный код.

4. Почаще тренируйтесь.

Профессионал всегда лучше любителя. Делайте из себя профессионалов.

5. Главное победа, но участие – тоже неплохо.

Отдыхайте. Приезжайте на контест, как на праздник. Старайтесь расслабиться и получить максимум удовольствия.

**P.S.** На одном этаже с моим кабинетом нет кабинета Дона Кнута. Я не ем креветок с видом на Тихий Океан. Но я трижды выиграл чемпионат Урала и взял бронзу на чемпионате мира, хотя никогда не показывал особенных успехов в школьных олимпиадах и не могу похвастаться выдающимися способностями. Когда-то мне сложно было даже и понять – почему это произошло. Но задним умом все сильны. Теперь-то я понимаю: я просто старался следовать тем принципам, которые описаны выше.

*Я не ем креветок с видом на Тихий Океан.*

***Шамгунов Никита Назимович, чемпион Урала 1998–2000 гг., программист Transas Software House, Санкт-Петербург.***

***Волков Леонид Михайлович, чемпион Урала 1999–2001 гг., руководитель лаборатории Интернет-технологий компании «СКБ Контур», Екатеринбург.***