Опубликовано в журнале «Телекоммуникации и информатизация» образования. 2003, №1 .– с. 61–73.

Г.А. Корнеев

Р.А. Елизаров

Автоматическое тестирование решений на соревнованиях по программированию.

Основной целью данной статьи является описание автоматического тестирования программ на соревнованиях по программированию. Где применяется подобное тестирование?

В настоящее время широкое распространение получили различные соревнования по программированию, например, этапы Командного студенческого чемпионата мира по программированию (ACM ICPC), Всероссийские командные олимпиады школьников по программированию. Кроме того, указанные задачи встречаются в современных технологиях дистанционного обучения, в частности, в проекте Интернет–школы программирования (Internet Programming School, IPS), реализуемом на кафедре компьютерных технологий СПбГИТМО (ТУ).

Так же в данной статье описывается новый подход к тестированию задач, основанный на игровой стратегии. Это означает, что вместо традиционного тестирования на заранее определенном наборе тестов наша система “играет” с проверяемой программой, и по результатам данной “игры” определяет общий результат. Данный метод позволяет адекватно оценивать задачи, для которых оптимальное решение практически не достижимо в силу высоких требований к вычислительным ресурсам. Кроме того, позволяет тестировать управляющие программы, когда автоматическая тестирующая система эмулирует управляемый объект и оценивает качество управления.

Наконец, универсальность нового подхода позволяет настраивать систему проверки решений задач на работу в режимах тестирования, регламентируемых различными требованиями, которые определяются согласно правилам конкретного соревнования. Например, весьма различаются условия проверки результатов, принятые в упомянутых выше соревнованиях, но и в личных соревнованиях школьников России, а также в соревнованиях, где участники сдают не программы, а только результаты их работы.

В настоящее время, автоматическое тестирование программных решений, обсуждается достаточно широко. На данный момент, разработаны и успешно используются различные методики автоматического тестирования и программные продукты, основанные на их применении. Но большинство таких продуктов “заточены” под проведение соревнований с

конкретным регламентом и практически не подлежат изменению для поддержки соревнований с существенно отличающимися регламентами проведения.

Так же, в последнее время, все большую популярность приобретают соревнования реального времени, вытесняющие классические соревнования с отложенной проверкой. Но соревнования такого рода требуют значительно более высокого уровня организации проверки решений участников, так как время, выделяемое на тестирование решения, измеряется минутами. Напротив, в соревнованиях с отложенной проверкой время тестирования решения практически не ограничено.

Описанный в данной статье методы тестирования был успешно апробирован при проведении реальных соревнований (четверть и полуфинала Командного студенческого чемпионата мира по программированию), а так же используется в тестирующем ядре интернет-школы программирования [8] и системе автоматического проведения интеренет-соревнований [7]. Более подробно об этих проектах можно узнать в работе [1]

Точное определение используемых терминов и аббревиатур приведено в соответствующем разделе в конце статьи.

Обоснование необходимости автоматизации процесса тестирования

На данный момент, практически отсутствуют статьи, посвященные системам автоматического тестирования задач, хотя в разных странах созданы и успешно действуют несколько таких систем. Но работать, с большей их частью, не говоря уж о расширении возможностей, могут только авторы.

В то же время, проводится все больше различных соревнований по программированию, на большинстве которых участники, как результат своей работы сдают исходные тексты программ. В дальнейшем эти программы необходимо как-то оценивать.

При небольших масштабах соревнования (участники сдают несколько десятков решений) часто применяются методы ручного тестирования. В этом случае, член жюри самостоятельно компилирует каждое представленное решение, запускает его на последовательности тестов из заранее определенного (но не обязательно известного участникам!) набора, и, наконец, оценивает результат, выданный программой на каждом из них.

Данный подход сопряжен с большими затратами времени и ручного труда. В частности, как показывает опыт, неверно написанные программы участников часто мешают нормальной работе компьютера проверяющего, вследствие чего, последний вынужден производить частые перезагрузки, что еще больше увеличивает временные затраты. Очевидно, данный подход неприменим при более интенсивной нагрузке.

Еще одним недостатком данного подхода является невозможность корректно оценить время, которое будет потрачено на проверку решения участника (так как “хорошее” решение быстро пройдет все тесты, а “плохое” будет вынуждать производить перезагрузку после каждого запуска). Данный недостаток практически не позволяет использовать описанный подход для проведения соревнований реального времени. Несмотря на описанные недостатки, данный метод до сих пор применяется на соревнованиях низкого уровня, таких как, школьные и районные олимпиады школьников по программированию.

Единственным способом преодоления недостатков описанного подхода является автоматизация различных этапов тестирования решений.

Наиболее часто автоматизируемым этапом, является компиляция решений участников. Но, на практике, автоматизация данного этапа не решает ни одну из проблем, присущих ручному тестированию.

Серьезным шагом является автоматизация запуска решений участников на тестах. Обычно, для этого создается программный комплекс, который ограничивает действия, которые может производить решение участника (обычно такую ограниченную среду называют “песочница” — sandbox). В частности, строго ограничивается время и количество памяти, используемое тестируемой программой. Это позволяет ускорить процесс тестирования в несколько раз, а главное — существенно более точно оценивать время, необходимое на тестирование одного решения участника, что позволяет использовать такую тестирующую систему в соревнованиях реального времени. Такая автоматизация позволяет проверять за разумное время, уже сотни, а не десятки решений.

Тестирующие системы данного типа успешно используются даже на соревнованиях такого уровня, как финал Командного студенческого чемпионата мира по программированию. Но, так как проверка правильности результата, выданного решением участника на тесте, выполняется вручную, то данному методу так же присущи существенные недостатки.

Одним из них является практически полная невозможность проверки задач, с неоднозначным решением, в то же время, задачи такого типа часто встречаются на практике. Нередко, для обеспечения единственности решения, в задачу вносятся определенные ограничения, например, полученное решение, должно быть первым в лексикографическом порядке. Данные ограничения обычно не влияют на алгоритмическую сложность задачи, но зачастую существенно увеличивают ее техническую сложность, что, очевидно, является негативным фактором.

Вторым недостатком частичной автоматизации является необходимость присутствия достаточно квалифицированного человека — а во многих случаях, и группы людей — для выполнения неавтоматизированных этапов, что практически не позволяет создавать системы,

работающие круглосуточно. При этом работа, выполняемая человеком, весьма рутинна, из-за чего со временем существенно снижается производительность труда и надежность работы исполнителя. Таким образом, возможности использования частично автоматизированных проверяющих систем ограничены проведением соревнований продолжительностью не превышающих несколько часов. Так же практически исключено проведение дистанционных и интернет-соревнований, как следствие большого числа запросов на проверку.

Итак, чтобы исключить ошибки, более точно — для того, чтобы все участники были в равных условиях, а также для создания тестирующих систем, рассчитанных на круглосуточное использование, необходима полная автоматизация процесса тестирования. В частности, роль администратора такой системы сводится к добавлению новых задач, доступных для тестирования, и, возможно, расширению доступных вычислительных ресурсов.

При этом автоматические системы тестирования могут быть применены во всех типах соревнований, в том числе, для круглосуточных соревнований реального времени. Они могут тестировать тысячи решений в час, что существенно для проведения интернет-соревнований.

Еще одним плюсом автоматических систем тестирования является возможность тестирования задач с существенно не единственным решением, а так же оптимизационных задач. Более того, при использовании игровой стратегии, возможно тестирование задач, для которых точное решение не известно, что существенно увеличивает круг возможных задач.

Кроме того, полная автоматизация системы тестирования позволят тестировать существенно большее количество программ участников за отведенное время. При проведении реальных интернет–соревнований системе иногда приходится тестировать десятки программ в минуту, что, очевидно, невыполнимо ни при ручном, ни при частично автоматизированном тестировании.

Обзор существующих систем

Рассмотрим примеры существующих систем автоматического тестирования, успешно используемых для проведения различных соревнований.

Такие системы, распадаются на два практически не пересекающихся класса — системы проведения интернет-соревнований и системы проведения очных соревнований. Существенное отличие первых от вторых заключается в том, что они работают круглосуточно, практически, без вмешательства администратора, а системы проведения очных соревнований используются только непродолжительный период времени, во время проведения соревнования как такового, и постоянно находятся под наблюдением администратора.

Системы проведения интернет- соревнований

Рассмотрим основных представителей класса систем проведения интернет-соревнований и архивов задач с автоматической проверкой. Такие системы уже существуют как в Росси, так и за рубежом.

Существенным недостатком дистанционных соревнований является их временная “привязка” к очным соревнованиям, что практически исключает возможность участия лиц, разница по времени с которыми превышает 8-10 часов. В то же время, дистанционные олимпиады обычно проводятся вместе с очными соревнованиями достаточно высокого уровня и участвовать в них не подготовленным участникам не интересно, так как предлагаемые задачи слишком сложны. Системы проведения интеренет-соревнований обычно позволяют не только участвовать в различных дистанционных соревнованиях, но и создавать собственные соревнования, необходимой сложности и проводимые в требуемое время, что существенно повышает интерес к ним. Так же для них имеется возможность участвовать в одном соревновании несколько раз, с некоторым временным интервалом, что позволяет участникам сравнивать результаты своих выступлений и отслеживать свой прогресс. Опыт показывает, что многие участники теоретически знают, как решать задачу, но не могут корректно реализовать имеющийся алгоритм на выбранном языке программирования. Таким образом, наравне с оценкой алгоритмов участники имеют возможность проверить качество кодирования.

Системы проведения интернет-соревнований требуют постоянной поддержки и пополнения набора задач, доступных для тестирования. По этому, системы автоматического тестирования данного класса мало распространены. Фактически эффективно поддерживать такую систему могу только организации с развитой инфраструктурой проведения соревнований по программированию, имеющие солидную базу для подготовки победителей соревнований самого высокого уровня. Только несколько российских и зарубежных ВУЗов удовлетворяют данным требованиям, что существенно сужает круг претендентов на создание такой системы.

Еще одной проблемой является написание надежной системы автоматического тестирования, которая будет успешно сопротивляться попыткам нарушить ее работу. Разрешение данной проблемы самой по себе является весьма не тривиальной задачей.

Рассмотрим подробнее некоторых представителей систем проведения интеренет-соревнований.

Valladolid Programming Contest Problem Set [5] — Старейший архив олимпиадных задач. На данный момент, является одним из самых популярных сервисов такого рода в Интернет. Содержит сотни (!) задач с тестами. К сожалению, все общение с тестирующей системой происходит по e-mail, что является существенным недостатком для некоторых

пользователей. Еще одним недостатком данной системы является использование только операционной системы Linux, и соответствующих языков программирования (GNU C, GNU C++, Free Pascal), что существенно сужает количество возможных участников. Для каждой задачи подводится статистика — рейтинг сложности (отношение количества правильных решений задачи к общему количеству попыток сдать данную задачу), что позволяет участникам выбирать посильные для них задачи. Архив активно пополняется задачами с различных соревнований, как очных, так и проводимых через Интернет.

Ural State University Problem Set with Online Judge System [6] — Архив задач и система проведения дистанционных олимпиад. Данная система популярна в России и Китае. Дистанционные олимпиады обычно проводятся одновременно с очными олимпиадами Уральского Государственного Университета. О проведении соревнования участники предупреждаются по e-mail, примерно за неделю до его начала. Обычно в соревнованиях такого рода участвуют 200-300 команд. Для подсчета результатов используются правила ACM ICPC. Для участников доступен стандартный набор языков программирования: C, C++ и Pascal (Delphi). К сожалению, данная система автоматического тестирования допускает только тестирование зада с однозначным ответом (выходные данные, сформированные программой участника, сравниваются с эталонными).

Системы проведения очных соревнований

Системы проведения очных соревнований специально предназначены для очных олимпиад и не включают возможности проведения даже дистанционных соревнований.

Рассмотрим конкретные примеры автоматических проверяющих систем этого класса.

NPC2 (Антон Суханов, Роман Елизаров) — Одна из первых систем автоматического тестирования. Предназначена для проведения соревнований, с использованием сети Novel Netware, позволяет тестировать только программы написанные под DOS. Данное ограничение является существенным, и для его устранения необходимо переписывать тестирующую систему заново. При проверке, нагрузка может быть распределена на несколько компьютеров, связанных в локальной сетью. Данная автоматическая система тестирования была успешно использована для проведения многочисленных соревнований по программированию. Для написания проверяющих программ для этой системы была разработана библиотека Testlib, различные модификации которой, в данное время, используются практически во всех российских системах автоматического тестирования. Исходно, система предназначалась для проведения соревнований по регламенту ACM ICPC, в последствие она была адоптирована для проведения Всероссийских Олимпиад Школьников по Информатике.

Cyber Judge (Максим Бабенко) — Относительно новая система автоматического тестирования, предназначенная для проведения соревнований по регламенту Всероссийских олимпиад школьников по информатике. Особенностью данной системы является возможность не только автоматического, но и ручного тестирования решений, что является существенным для ROI, так как по правилам, тестирование происходит в присутствии участников. Данная система поддерживает проверяющие программы, написанные при помощи Testlib, но не совместима с NPC2 по формату конфигурационных файлов. Таким образом, перенос задач из одной системы в другую, сводится к изменению конфигурационных файлов задачи. Данная система находится в стадии модификации, и в будущем, возможно, будет поддерживать проведение дистанционных соревнований.

Автоматическое тестирование

Задача процедуры тестирования программного решения состоит в определении, решает ли тестируемая программа требуемую задачу.

В последнее время широкое применение находят методы, основанные на анализе исходного кода программ. Но для любого достаточно сложного языка программирования (эквивалентного машине Тьюринга), существуют программы, для которых невозможно эффективно определить, не только решают ли они заданную задачу, но даже, закончит ли программа выполнение за конечное время.

Так как, точное определение корректности программы невозможно, то широкое применение находят различные эмпирические методы тестирования. При этом в зависимости от структуры задачи используются различные методы тестирования.

Назовем метод тестирования **косвенным**, если для определения корректности ответа, получаемый на наборе входных данных, производится проверка принадлежности его некоторому, заранее вычисленному множеству.

Косвенные методы обычно применяются при тестировании задач с единственным решением, а так же задач, для которых известны эффективные методы получения всех возможных правильных ответов.

Недостаток косвенных методов состоит в том, что для проверки правильности результата мы должны уметь решать исходную задачу, или даже более сложную (в случае, когда решение исходной задачи не единственно).

Назовем метод тестирования **прямым**, если для проверки правильности результаты нет необходимости иметь решение исходной задачи.

Рассмотрим следующую классификацию задач:

1) По количеству правильных ответов.

A) Задачи с единственным ответом.

B) Задачи с конечным множеством правильных ответов.

C) Задачи с бесконечным множеством правильных ответов.

2) По существованию эффективного метода решения

A) Для задачи известен эффективный метод получения всех ответов.

B) Для задачи известен эффективный метод получения некоторого ответа.

C) Для задачи не известны эффективные методы решения.

3) По существованию эффективного метода проверки решения, не включающего решение исходной задачи.

A) Для задачи известен эффективный метод проверки, не требующий решения исходной задачи.

B) Такой метод не известен.

Будем обозначать типы задач трехбуквенными сокращениями, где каждая буква является значением соответствующего параметра. На пример, задача типа BCA — имеет конечное множество решений, для которого не известны эффективные методы нахождения хотя бы одного элемента, но существует эффективный метод проверки принадлежности данному множеству.

Очевидно, что не все возможные 18 типов задач имеют смысл. Рассмотрим примеры задач для каждого типа.

AAA. Задача извлечения арифметического квадратного корня из целого числа, являющегося полным квадратом. Для данной задачи существует метод решения линейный по количеству цифр в исходном числе. Так как ответ единственен, данным методом мы получаем все возможные решения. Посредством возведения полученного решения в квадрат, и сравнения с исходным числом производится эффективная проверка правильности результата, не основанная на решении исходной задачи.

AAB. Задача перемножения двух чисел. Очевидно, задача имеет единственное решение, которое можно получить эффективным методом. При этом единственным разумным методом проверки результата является перемножение исходных чисел.

ABA и ABB. Так, как правильно решение единственно, и мы можем эффективно получить некоторое правильное решение, то мы имеем эффективный метод получения всех возможных решений. Таким образом, не существует задач таких типов.

ACA. Задача вычисления дискретного логарифма, т.е. нахождение такого K, что $B = A^K \bmod N$. Если N — простое число, то результат единственен. А для его проверки нужно просто возвести число A в K-ю степень по модулю N.

ACB. Задача разложения числа на простые множители. Одним из этапов проверки решения является определение простоты числа, которая не может быть произведена эффективно.

BAA. Представление натурального числа N в виде суммы двух различных натуральных слагаемых. Мощность множества решений равна $\lfloor N/2 \rfloor - 1$. Проверка производится простым сложением полученных слагаемых.

BAB. Нахождение минимального пути в графе.

BBA. Нахождение гамильтонова пути в плотном графе.

BBB. Нахождение минимального пути во взвешенном графе.

BCA. Задача разложения составного числа на пару натуральных множителей, не равных единице. Задача имеет конечное множество правильных ответов, т.е. его мощность не превосходит некоторого числа. Но эффективного метода нахождения хотя бы одного решения не известно. При этом, если ответ найден, проверить его не составляет труда — необходимо просто перемножить полученные множители и сравнить результат с исходным числом.

BCB. Задача о назначениях.

CAA и CAB, так как все элементы бесконечного множества, не могут быть получены, то зада данных типов не существуют.

CBA. Задача нахождения Пифагоровой тройки (натуральных чисел A, B и C, таких, что $A^2 + B^2 = C^2$), все числа которой больше заданного.

CCA. Поиск простого числа, больше заданного.

CCB. Численное решение “Задачи трех тел”

По отношению к тестированию задачи разбиваются на три класса:

типы AAA, AAB, BAA, BAB, ACB допускают прямое тестирование.

типы AAA, ACA, BAA, BBA, BCA, CBA, CCA, допускающие косвенное тестирование.

типы BBB, BCB, CBB, CCB Задачи, не допускающие не прямых не косвенных методов тестирования. Для задач этого класса не существует эффективного метода проверки корректности ответа.

При этом, путем наложения дополнительных условий на ответ, задачи типов BBB и CCB зачастую удается преобразовать в задачи с единственным решением, что переводит их в класс задач, допускающих прямое тестирование.

Рассмотрим различные методы тестирования.

Тестирование на наборе тестов

При использовании данного метода тестирования программа считается корректной, если она выдает правильный результат на некотором конечном подмножестве множества всевозможных входных данных

Данный метод имеет как прямой, так и косвенный вариант, в зависимости, от способа проверки ответа.

В прямом варианте полученный ответ проверяется с помощью соответствующей процедуры, а в косвенном ответ сравнивается со всеми правильными ответами для данного набора входных данных (обычно применяется в случае единственности ответа).

Тестирование на наборе тестов

Программа считается корректной, если она выдает правильные ответы на заданном количестве случайных наборов входных данных.

Данный метод обычно используется как косвенный, в случаях, когда известен эффективный метод проверки правильности решения, но не нахождения решения как такового и может быть применен.

Тестирование посредством эмуляции

Программе на вход подается некоторая исходная ситуация, после чего она изменяется, в соответствии с полученным ответом и вновь подается на вход тестируемой программе. Если после ряда запусков программе удается добиться требуемого результата, она считается корректной.

Обычно используется для программ управления или программ для игр (в математическом смысле этого слова), в которых не известна оптимальная стратегия. В первом случае, тестирующая программа эмулирует управляемый объект, во втором, “играет против” тестируемой программы.

Все приведенные методы тестирования могут быть реализованы в виде “диалога” тестирующей и тестируемой программы, что позволяет реализовать единую тестирующую систему, включающую их. Данный подход и был заложен, как базовый в автоматическую тестирующую систему PCMS2.

Игровая стратегия тестирования

На данный момент подавляющее большинство систем автоматического тестирования использует по тестовый подход. В рамках данного подхода тестируемая программа запускается на заранее определенном наборе тестов, после чего выданный ею результат проверяется. Существенным недостатком данного подхода является невозможность адаптации системы тестов под конкретное решение, для определения его сильных и слабых сторон.

При использовании данной стратегии, проверка решения рассматривается как игра двух противников: тестирующей системы и тестируемой программы. Первый ход осуществляет тестирующая система (проверяющая программа). Она подготавливает входные данные, для запуска тестируемой программы, а так же устанавливает ограничения на количество ресурсов, которые будут доступны ей. Далее ход переходит к тестируемой программе. Она запускается на входных данных, подготовленных тестирующей системой, и выдает некоторый результат. При этом осуществляется контроль за объемом использованных ресурсов. Ход вновь получает тестирующая система. Она анализирует результат, полученный тестируемой программой. На основе результатов анализа принимается решение о продолжении тестирования. Если тестирование продолжается, то проверяющая система подготавливает новый набор входных данных, при этом может быть использована информация, полученная от тестируемой программы, на предыдущих шагах. После чего ход снова передается тестируемой программе, и так далее.

Игровая стратегия тестирования позволяет повысить гибкость проверки решений и существенно расширяет круг задач, доступных для тестирования.

Очевидно, по тестовый подход является подмножеством предложенной стратегии тестирования. При этом, информация, полученная при запуске программы участника на одном тесте, ни как не используется для проверки других.

То, что, в свой ход, автоматическая система тестирования может производить произвольные действия (в том числе, и запуск внешних программ), позволяет без дополнительных механизмов осуществлять вероятностное тестирование. При этом, тестирующая система, создает случайный набор входных данных, удовлетворяющих условию задачи и передает ход тестируемой программе. Для проверки полученных результатов возможны следующие подходы:

- Тестирующая программа непосредственно проверяет правильность полученного ответа. Применяется при тестировании задач классов ??A.

- Запускается правильное решение (решение жюри). Результаты, выданные тестируемой программой и правильным решением сравниваются. Применяется для тестирования задач с единственным правильным решением.
- Запускается генератор множества всех правильных решений. Результат, выданный тестируемой программой, проверяется на принадлежность сгенерированному множеству. Применяется для тестирования задач класса ?A?, при не большой верхней границе мощности множества правильных ответов.
- Запускается решение жюри и полученный результат, сравнивается по оптимальности с результатом, выданным тестируемой программой. Данный подход используется для тестирования оптимизационных задач, для которых точный ответ получить не представляется возможным (на пример, задачи Коммивояжера для плотных графов с большим числом вершин).

Использования последнего из предложенных подходов, для тестирования на заранее заданном наборе тестов, позволяет осуществлять детерминированное тестирование, задач, точное решение для которых не известно.

Дополнительной возможностью игровой стратегии тестирования является сравнительное тестирование игровых задач. При этом проверяющая программа действительно играет с тестируемой. В начале, задается исходная позиция в игре, после чего, одна из сторон делает первый ход и передает ход другой стороне, которая в свою очередь делает свой ход и т.д. При этом проверяющая программа проверят корректность производимых ходов обеих сторон, и оценивает результаты игры.

Так же игровая стратегия тестирования позволяет осуществлять проверку программ управления. При этом, проверяющая система, в свой ход, проверяет корректность управляющих воздействий, предложенных тестируемой программой и рассчитывает отклик управляемой системы на произведенные воздействия, после чего подает полученный результат на вход тестируемой программе и т.д. Тестируема программа, признается корректной, при достижении требуемого состояния управляемой системы.

Таким образом, в рамках игрового подхода к тестированию, может быть осуществлен двухуровневый подход к тестированию. На верхнем уровне используется заранее заданный набор тестов. На нижнем уровне производится несколько запусков тестируемой программы, с использованием схемы сравнительного тестирования или тестирования программ управления. На верхнем же уровне, оценивается результат произведенной последовательности запусков в целом (результат игры, состояние управляемой системы, и т.п.).

Описанные в данном разделе новые методы тестирования не могут быть реализованы в рамках классической по тестовой стратегии.

Используемые термины и сокращения

В данном разделе описаны термины, используемые других частях настоящей статьи. При этом смысл многих терминов сужен, по сравнению, с их обычным смыслом. Это связано, с тем, что в статье, главным образом, рассматривается тестирование задач на соревнованиях по программированию.

Автоматическое тестирование (Automated judging) — новая концепция подхода к тестированию задач, основанная на полностью автоматизированном выполнении всех действий, необходимых для проверки решения, сданного участником соревнования, не требующем, но и не исключающем вмешательство человека. В частности, автоматическое тестирование может происходить круглосуточно, в соответствии с запросами пользователей системы автоматического тестирования, реализующей данный подход.

Автоматическая система тестирования (Automated judging system) — комплекс программно-аппаратных средств, реализующих концепцию автоматического тестирования.

Задача (Problem) — Не формальное, но полное описание зависимости выходных данных от входных данных. Так же в условии задачи, обычно указывается максимальное время работы правильного решения на одном тесте, и максимальное количество памяти, которое может использовать решение участника.

Подход, решение участника (Run) — попытка участника сдать задачу, с целью получить положительный результат проверки. Автоматическая система тестирования должна проверить решение участника, и сообщить ему полученный результат. В случае соревнований реального времени, так же обновляется таблица текущих результатов.

Тест (Test) — единица тестирования задачи, в большинстве автоматических систем тестирования. Содержание теста определятся до процесса тестирования. Набор тестов для каждой задачи не изменен в течение всего соревнования. Тест может быть пройденным (в случае, если, решение участника не выполнило недопустимых операций, и выданный результат корректен), и не пройденным, в противном случае.

Проверяющая программа (Tester) — программа, определяющая правильность результата, выданного программой участника на некотором тесте. Обычно на вход проверяющей программы подаются несколько файлов: входной файл, полученный решением участника, выходной файл — результат, сформированный решением, а так же, файл, содержащий информацию помогающую проверить данный тест (если ответ задачи единственен, данный файл содержит его).

Результат проверки (Outcome) — общий результат проверки задачи. Зависит от регламента проведения соревнований, может быть как простым: зачтено / не зачтено, так и более сложным, например, количество очков, полученных участником за данную задачу, или содержать в себе причину, вызвавшую отказ в зачтении задачи.

Администратор — лицо (или группа лиц), ответственное за поддержание системы автоматического тестирования в рабочем состоянии, добавление задач и другие работы по ее обслуживанию.

Соревнование (Contest) — в рамках данной работы, соревнование определяется правилами его проведения (регламентом), набором задач, доступных для решения, временем и местом проведения соревнования. Результатом соревнования является итоговая таблица, характеризующая рейтинг участников, по определенной шкале.

Регламент соревнования (Contest rules) — порядок регистрации участников, проведения соревнования, правила подсчета результатов в целом, и по каждой задаче в отдельности, допустимые к использованию языки и средства программирования. В данной работе регламент соревнования рассматривается только в части, относящейся к правилам оценки одной задачи, так как задача является базовым объектом тестирования решения участника, в частности, набор языков программирования, допустимых для ее решения. Описание различных регламентов проведения соревнований можно найти в [3] и [4], а так же в других разделах данной работы.

Участник, Команда (Participant, Team) — Лицо, участвующее в соревновании (группа лиц, участвующих в соревновании как единое целое). Команда характеризуется в таблице результатов одной строкой, в не зависимости, от реального количества лиц ее составляющего. Правила отбора участников, и комплектования команд определяются регламентом соревнования. Так как, для отдельных членов команды результаты не подводятся, данной работе участники и команды не разделяются, и для их обозначения используется слово “участник”.

Соревнование реального времени — соревнование, по регламенту которого участники могут сдавать решения во время проведения соревнования, и должны получать соответствующие результаты в течение оговоренного периода с момента сдачи решения на проверку. Примером таких соревнований могут служить различные этапы Командного студенческого чемпионата мира по программированию.

Соревнование с отложенной проверкой — соревнования, по регламенту которых решения проверяются по окончанию соревнования. Примером соревнований данного типа являются различные этапы Всероссийской олимпиады школьников по информатике.

Интернет–соревнование — соревнование, проводимое посредством глобальной сети Интернет, при этом, участники соревнования могут находиться в произвольной точке Земли. Специфика данных соревнований допускает весьма значительное количество участников, а также возможность проведения круглосуточных соревнований.

Очное соревнование — соревнование, участники которого непосредственно присутствуют в зоне проведения соревнования. В этом случае их количество ограничено существующей технической базой и известно до проведения соревнования.

Дистанционное соревнование — очное соревнование, одновременно проводимое в глобальной сети Интернет. При этом, участники, использующие Интернет, видят результаты участников очного соревнования, но не наоборот. Данный вид соревнований приобретает все большую популярность, так как для их проведения, практически не требуется дополнительная инфраструктура (по сравнению с очными соревнованиями).

APPES — Automated Programming Problem Evaluation System (Автоматическая система проверки задач по программированию). Предшественник PCMS2 Kernel.

PCMS2 — Programming Contest Management System v. 2 (Система управления соревнованиями по программированию, версия 2).

PCMS2 Kernel — Ядро PCMS2, осуществляющее автоматическую проверку решений участников.

ROI — Russian Olympiad in Informatics (Всероссийская олимпиада школьников по программированию).

IOI — International Olympiad in Informatics (Международная олимпиада школьников по программированию).

IPS — Internet Programming School (Интернет-школа программирования).

ACM ICPC — ACM International Collegiate Programming Contest (Студенческий командный чемпионат мира по программированию).

Список литературы

1. Казаков М.А. Разработка и внедрение системы поддерживающей новые технологии обучения программированию
2. Корнеев Г.А. Автоматизированная система тестирования программ. // Материалы VIII международной конференции "Современные технологии обучения <<СТО-2002>>". 24 апреля 2002 года. -- Том 2, с.327-329. -- СПб.: СПбГЭТУ, 2002.

Ресурсы глобальной сети Интернет

3. http://icpc.baylor.edu — официальный сайт Командного студенческого чемпионата мира по программированию
4. http://www.ioi2001.edu.fi — сайт Международной олимпиады школьников по программированию 2001 года
5. http://acm.uva.es — сайт в испанском городе Вальядолиде — первый архив задач с автоматической системой проверки
6. http://acm.timus.ru — сайт в городе Екатеринбурге — первый российский архив задач с автоматической системой проверки
7. http://neerc.ifmo.ru/online — архив соревнований с системой автоматического проведения online-соревнований
8. http://ips.ifmo.ru — Интернет-школа Программирования

ОБ АВТОРЕ

Корнеев Георгий Александрович — студент СПбГИТМО(ТУ), председатель жюри четвертьфинала и член жюри полуфинала Командного чемпионата мира по программированию 2001/2002 года.