

Инструкции участника муниципального этапа Всероссийской олимпиады по информатике

2016-2017 учебный год

1. Общая информация

Традиционная схема проведения олимпиад по информатике подразумевает выдачу участнику некоторого набора задач, в каждой из которых требуется разработать алгоритм переработки входных данных в соответствующие выходные и реализовать его в виде компьютерной программы на том или ином языке программирования, разрешённом к использованию на данном соревновании.

Проверка работ участников не включает изучение листинга написанных программ. Вместо этого программным комитетом предлагается набор тестов и ответом к ним. В случае, когда правильный ответ неединственный, предлагается программа анализа получаемого ответа на корректность. При этом проверка представляет собой запуск программ участника с подачей ей тестов соответствующей задачи по одному с дальнейшим анализом получаемого ответа. (В частности, в связи с этим требуется, чтобы программа считывала входные данные и записывала выходные точно в соответствии с указанным форматом.) Руководит этим процессом специальная программа (также поставляемая в составе пакета), называемая *проверяющей системой*.

В течение последних шести лет проверка проводилась посредством локальной проверяющей системы, рассылаемой в муниципальные жюри. Однако эта программа разрабатывалась под 32-битную версию операционной системы Microsoft Windows XP и к настоящему времени морально устарела, так как зачастую не может корректно проверять программы, работая под управлением современных операционных систем семейства Microsoft Windows. Жюри во многих территориях столкнулись с такими недостатками предлагаемой проверяющей системы на муниципальных этапах 2012-13 и 2013-14 учебно-го годов.

В прошлом учебном году предметной комиссией было принято решение предоставить жюри муниципальных этапов возможность проводить проверку решений школьников на системе Timus Online Judge. Опыт оказался достаточно успешным, поэтому в этом году эта практика будет продолжена. В связи с этим участникам из тех муниципалитетов, где при организации районного тура привлекается Timus Online, следует ознакомиться с инструкциями по его использованию (см. раздел 2).

Однако также будет подготовлен и пакет, основанный на имеющейся версии локальной проверяющей системы, в частности, для проведения апелляций. Поэтому имеет смысл ознакомиться и инструкциями по его использованию тоже (см. раздел 3).

2. Портал Timus Online Judge

2.1. Проведение соревнований

Для участия в соревновании на портале Timus требуется войти в раздел портала, посвящённый соревнованиям <http://acm.timus.ru/offline>. В имеющемся окне будет предложено ввести свой идентификатор участника. В рамках проведения муниципального тура будет открыто пять соревнований: пробный тур (с 10⁰⁰ 18 ноября 2016 года до 21⁰⁰ 4 декабря 2016 года) и четыре соревнования по одному для каждого класса, участвующего в соревновании (с 10⁰⁰ до 14⁰⁰ 5 декабря 2016 года). Отдельные соревнования для разных классов требуются потому, что в рамках одного соревнования набор задач фиксированный, а пакеты каждого класса отличаются друг от друга по составу задач. Задача

пробного тура — освоение участниками системы портала. Соответственно, участник видит только доступные ему соревнования: пробный тур и соревнование соответствующего класса.

После входа в соревнование участнику доступен список задач. Рядом с каждой задачей имеются ссылки «Отправить на проверку» (процедура приёма задачи на проверку описана ниже) и «Протестировать» (отправить задачу на сервер для прогона на примерах, но без дальнейшего запоминания для проверки на основном наборе тестов). В настоящее время условия задач для такого типа соревнований не отображаются на странице соревнования, их нужно получить отдельно в бумажном или электронном виде.

Так же, как и в соревнованиях прошлых лет, участник должен написать консольную программу (то есть программу, работающую в текстовом окне без всякого графического интерфейса). В отличие от соревнований прошлых лет для поддержания автоматической проверки программы участников при вводе-выводе должны работать *с консолью*, то есть организовывать ввод, как если бы он шёл *с клавиатуры*, а вывод — как если бы он шёл *на экран*. При этом так же, как и раньше, ввод и вывод должны подчиняться оговоренному формату, в частности, не должно быть реализовано вывода никаких приглашающих сообщений, а также ожидания нажатия клавиши в конце работы программы.

На странице соревнования также содержится ссылка на страницу отправки текста программы на сервер (ссылка «Отправить на проверку»). В этой форме участник выбирает язык, на котором написана программа, решающая соответствующую задачу. Сам текст программы может быть переслан либо посредством вставки листинга в текстовое окно, либо пересылкой файла (кнопка выбора файла находится ниже текстового окна). Последнее предпочтительнее, поскольку при копировании могут возникнуть проблемы (например, скопирован не весь текст программы).

Страница соревнования становится доступной с момента начала соревнования, заложенного в систему. После момента окончания соревнования страницы отправки решений становятся недоступными. После окончания соревнования в любой момент участник имеет возможность, воспользовавшись своим идентификатором, скачать решения, принятые к проверке. Также после окончания проверки участнику становятся доступны результаты проверки в виде суммы баллов за каждую задачу, а также результаты прогона каждого теста (эта возможность сейчас в разработке).

На портале используются следующие языки и компиляторы:

- Паскаль:
 - FreePascal 2.6;
 - PascalABC 2.2;
 - Borland Delphi 7.0, BDS 2006;
- C/C++:
 - Microsoft Visual C/C++ 2010;
 - GCC/G++ 4.9;
 - GCC C11/G++ C++11 4.9;
 - Clang 3.5 C++;
- Microsoft VB.Net 2010;
- Microsoft C# 2010;
- Java 1.8;
- Python:
 - версия 2.7;
 - версия 3.4;
- Ruby 1.9;

- Scala 2.11;
- Haskell 7.6.

Соответственно, для работы локально на машинах должны быть установлены необходимые среды из указанного списка. Также для проведения соревнования на компьютерах участников требуется *постоянное Интернет-соединение* для отправки решений на сервер проверки и получения вердиктов участниками.

После загрузки программы на сервер, она компилируется выбранным компилятором. Если компиляция проходит успешно, полученный исполняемый файл прогоняется на примерах из условия задачи и ответы сравниваются с эталонными. В случае неуспеха на том или ином этапе участник информируется (через окно браузера) о причине неуспеха. Если примеры пройдены успешно, программа принимается на проверку. Если было сделано несколько успешных посылок, то к проверке принимается версия программы, последняя из успешных.

Тестирование на основном наборе тестов проводится после окончания соревнования (или во время соревнования, в случае наличия свободных вычислительных узлов). Каждая задача снабжается некоторым количеством (5, 10, 20, 25, 50) равноценных тестов и оценивается из полного балла в 100 очков. Каждый тест оценивается независимо и при успешном своём прохождении приносит соответствующее количество очков. Сумма очков за задачу есть сумма очков за успешно пройденные тесты. Успешность прохождения теста — исполнение программы участника *с соблюдением ограничений по памяти и времени* (для задач пакета — 64 мегабайта и 2 секунды, соответственно) и совпадение результата работы с ответом программного комитета (в задачах с одним правильным ответом) или успешная валидация результата программой – проверщиком (в задачах с неединственным правильным ответом), также предоставляемой программным комитетом.

В процессе соревнования можно задавать вопросы программному комитету. Для этого на странице соревнования есть ссылка «Задать вопрос жюри». Ответы будут приходить на страницу, открываемую по ссылке «Сообщения от жюри(N)» (N — количество непрочитанных сообщений). Также на эту страницу приходят широковебательные сообщения от жюри. К сожалению, автоматической проверки наличия сообщений нет, поэтому разумно держать эту страницу открытой в браузере и обновлять её время от времени.

Краткое руководство по использованию портала можно прочитать на странице

<http://acm.timus.ru/help.aspx?topic=judge>.

(ссылка «Руководство» из раздела «Задачи»верху любой страницы портала).

2.2. Проведение апелляций

Важной частью всякого соревнования является апелляция, когда участники, недовольные вынесенным вердиктом, имеют возможность урегулировать своё мнение с жюри.

Традиционная форма апелляции на соревнованиях по информатике на высоком уровне (заключительный этап, Международная олимпиада) заключается в следующем. После окончания тура (и после того, как участники пообедают и отдохнут) проводится разбор задач тура, участникам объясняется, какие идеи были заложены в задачи, какие пути решения подразумевались, какие типовые ошибки могли иметь место, какие идеи закладывались в конкретные тесты. Затем после окончания проверки (в идеале — сразу после окончания разбора) участники ознакамливаются с результатами проверки, как с суммарным баллом по каждой задаче, так и с результатом прогона каждого теста. После этого у каждого участника есть возможность вернуться за своё рабочее место и, получив файлы тестов и верных ответов, самостоятельно прогнать те тесты, к проверке которых у него есть претензии. Если после самостоятельного ознакомления с поведением программы на тех или иных тестах претензии не сняты, подаётся апелляционное заявление и проводится

общение с жюри, в результате которого тест может быть зачтён (хотя такое происходит достаточно редко).

К сожалению, в рамках муниципального этапа такая схема в полном объёме не может быть реализована. В частности, портал Тимус не поддерживает прогон решения по запросу от участника. Поэтому для проведения апелляции предлагается следующая схема.

После окончания соревнования участникам рассылается ссылка на архив с локальной проверяющей системой (см. раздел 3) с паролем к этому архиву. Участники самостоятельно скачивают этот архив и знакомятся с разборами и идеями тестов, предлагаемыми в текстовом виде. Также они имеют возможность получить с Тимуса свои решения, принятые к проверке, и, имея тесты и локальную проверяющую систему, самостоятельно прогнать свои решения (с учётом ограничений локальной системы — см. раздел 3). После окончания проверки участникам также доступны их собственные результаты (на портале, а также разосланные по электронной почте).

Соответственно, если после такого ознакомления с результатами проверки у участника остаются вопросы, он вправе принять участие в процедуре апелляции, проводимой с участием локального жюри. При этом процедура апелляции представляет собой такой же прогон программы участника на локальной машине. При этом следует учесть, что при проверке на машине жюри нужно использовать тот же компилятор (или интерпретатор), который использовался участником. Например, неправильно использовать Python 3.x, если участник работал на Python 2.7 — программа, скорее всего, просто не запустится. Или неправильно использовать Borland Delphi, если участник писал на FreePascal — программа, скорее всего, не скомпилируется. Кроме того, следует уделить внимание тому, чтобы проверяющая машина имела вычислительную мощность, примерно равную мощности сервера, на котором работает портал Тимус:

- процессор: Core 2 Quad Q6600 (частота 2400 МГц, кэш второго уровня 8 МБ, частота шины 1066 МГц);
- память: 4 Гб;
- жёсткий диск: Seagate Barracuda 7200.10, 250 Гб;

Это необходимо для корректного оценивания выполнения программой ограничений по времени. В принципе, при условии установленной ОС Windows XP соответствующей мощностью обладает любой более или менее современный нетбук.

3. Локальная проверяющая система

Вместе с возможностью использовать портал Тимус для проведения муниципального тура предоставляется и локальная проверяющая система, аналогичная тем, которые предлагались в прошлые годы. Как было отмечено в разделе 1, система обладает рядом ограничений при своём использовании. Первое — невозможность проверки 64-битных приложений и программ на скриптовых языках, запускаемых с использованием 64-битных интерпретаторов. Второе — невозможность проверки .Net-приложений под операционными системами Microsoft Windows Vista/7/8/8.1/10. Третье — некорректная работа на виртуальной машине.

3.1. Состав проверяющей системы

Архив `testSys.rar`, содержащий версию проверяющей системы, настроенной для проверки программ, решающих задачи муниципального этапа олимпиады по информатике, будет размещён в сети Интернет. Ссылки на него будут разосланы за несколько дней до тура. Для сохранения решений неизвестными архив создаётся с шифрованием, пароль будет разослан после окончания соревнования.

Архив содержит три папки:

- **Tasks** — условия задач;
- **Solutions** — описания идей решений задач и тестов, предложенных программным комитетом для проверки работ;
- **TestSys** — проверяющая система.

Проверяющая система предназначена для использования под операционными системами семейства Microsoft Windows. В состав проверяющей системы входят исполняемый файл `!test.exe` и наборы тестов для всех задач всех пакетов. Тесты распределены по папкам 08, 09, 10, 11 для пакетов соответствующих классов. Папка каждого пакета содержит пять подпапок с тестами для каждой из задач пакета. Имя каждой такой подпапки включает в себя номер задачи и условное название задачи.

Папка каждой задачи содержит следующие файлы и папки:

- `!run-c.bat` — файл, запускающий проверку указанной программы (см. ниже) на всех тестах данной задачи, предназначенный для программ, написанных с использованием консольного ввода-вывода;
- `!run-f.bat` — файл, запускающий проверку указанной программы (см. ниже) на всех тестах данной задачи, предназначенный для программ, написанных с использованием файлового ввода-вывода;
- `*-cor.cpp`, `*-cor.exe` — файл исходного кода и исполняемый файл проверки файла теста на соответствие условиям задачи;
- `!samples.ini`, `!samp_cor.bat`, `!samp_val.bat` — файлы, связанные с файлами примеров из текста задач: описанием состава примеров и методов их проверки, файлы запуска проверки корректности входных данных примера(-ов) и правильности их ответов;
- `!tests-c.ini` — файл с описанием состава тестов для автоматической системы, предназначенный для программ, написанных с использованием консольного ввода-вывода;;
- `!tests-f.ini` — файл с описанием состава тестов для автоматической системы, предназначенный для программ, написанных с использованием файлового ввода-вывода;;
- `*_da.cpp`, `*_sk.pas`, `*_es.py`, `*_sk.hs` — файлы решений членов программного комитета на языке C++ (в стандарте C++11), Delphi Pascal, Python, Haskell;
- могут присутствовать файлы `check.cpp` и `check.exe`; это вспомогательные файлы валидации ответа программы для задач, в которых правильный ответ может быть неединственным;
- подпапка `samples` — папка с файлами примеров из текста задачи и ответов к ним;
- подпапка `tests` — папка с файлами тестов (имена, содержащие числа), файлами ответов (числовые имена с расширением `.ans`). Также могут присутствовать исходные тексты программ генерации случайных тестов и вспомогательные файлы (в частности, файлы с расширением `.t`), использовавшихся для генерации тестов.

Отметим, что все служебные файлы (для проверки корректности тестов, для валидации ответов, для генерации тестов), подготовленные программным комитетом, написаны на языке C++ в системе Microsoft VisualC++ 2013 (совместимо с версиями 6.0, 2003, 2005, 2008, 2010, 2012).

После получения пароля к архиву проверяющую систему следует распаковать в удобное для проверяющего место на жёстком диске компьютера, используемого для проверки.

3.2. Описание процедуры проверки

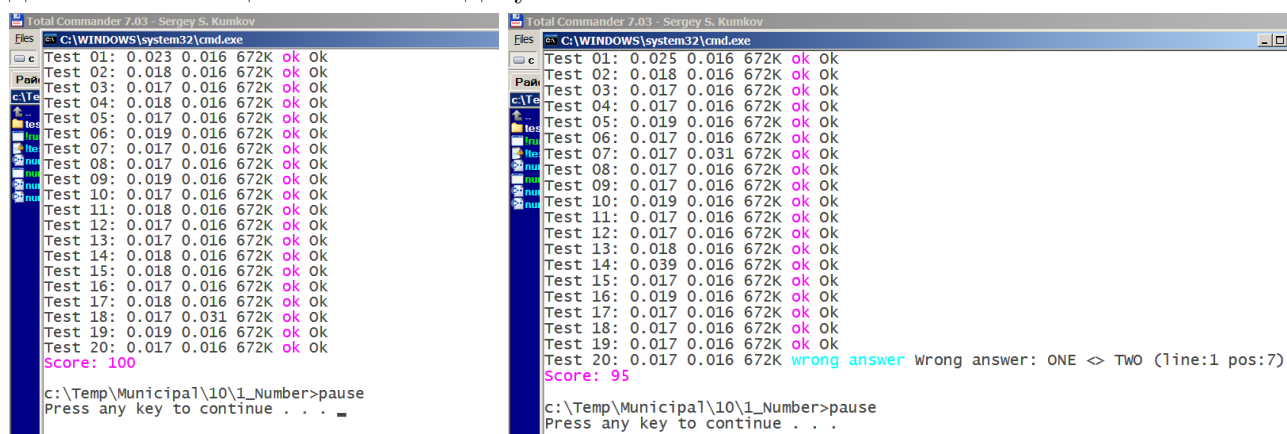
Для того, чтобы проверить ту или иную программу, следует скопировать её исполняемый файл (exe-файл) в папку соответствующей задачи, после чего в командной стро-

ке набрать `!run-c.bat <имя программы>` или `!run-f.bat <имя программы>` — в соответствии с тем, с каким вводом работает программа, консольным или файловым (например, `!run-c.bat Ivanov_1.exe`). Командную строку можно получить в бесплатном файловом менеджере FAR или в окне консоли, запускаемой Пуск → Выполнить; ввести `cmd`, нажать Enter. После запуска консоли нужно перейти на диск, где расположена проверяющая система, при помощи команды `<имя_диска>:`, а затем — в папку проверяющей системы, команда `cd <имя_диска>:\<путь_к_папке>`.

Для запуска на проверку программ, написанных на скриптовых языках (Python, Perl, Ruby), а также на языке Java, в файл запуска проверки нужно передать интерпретатор соответствующего языка и имя программы:

```
!run-c.bat "c:\Program Files\Java\jdk1.8.0_40\bin\java.exe" -jar Ivanov_1.jar
!run-c.bat c:\Python35\python.exe Ivanov_1.py
```

После запуска файла проверяющей системы он запустит проверяемую программу со всеми тестами задачи. При прогоне каждого теста на вход программы будет подан файл с данными этого теста. Запуск производится с контролем времени исполнения и объёма используемой памяти. Результат работы программы будет получен с выхода программы и проанализирован на правильность (сравнение с ответом в случае одного правильного ответа или валидация при помощи соответствующей программы в случае многих правильных ответов). В результате будет выведена диагностическая информация о результатах каждого теста и общего балла за задачу:



На левом снимке экрана приведён вид окна при полностью правильном решении, справа — с ошибками в некоторых тестах. Рядом с каждым неверно пройденным тестом будет приведена диагностика некорректности исполнения:

- **wrong answer** — неправильный ответ и, возможно, диагностика ошибки;
- **output file was not created** — работа программы завершилась без выдачи какой-либо информации в выходной поток;
- **memory limit exceeded** — программа использует памяти больше, чем оговорено в условии (64 мегабайта);
- **time limit exceeded** — программа выполняется дольше, чем оговорено в условии (2 секунды на тест);
- **idleness limit exceeded** — программа ожидает незапланированного ввода с клавиатуры;
- **crash** — программа некорректно завершила исполнение (деление на ноль, выход за границы массива, попытка некорректной работы с файлами и т.п.);
- **wrong output format** — нарушен формат выходных данных, проверяющая система не может разобрать данные выходного файл; также может выдаваться, если программа не выдала ничего.

В последней строке вывода (**Score:**) указан суммарный балл за задачу с учётом стоимости каждого теста и успешности прохождения тестов.