Принцип работы программы

Проверочная система состоит из двух компонентов:

- **Testing System** – непосредственно проверяющий модуль, занимается компиляцией, управляет запуском и проверкой результатов.
- **Report Builder** – генератор отчетов по результатам работы проверочной системы.

Описание основных файлов и каталогов и их назначение

contests – основной каталог, содержит подкаталоги с различными олимпиадами (описание данных каталогов ниже).
report – шаблон для генерируемых отчетов, как правило, его редактирование не понадобится.
ts.exe – проверочная система Testing System
rb.exe – генератор отчетов Report Builder

Содержимое каталога с олимпиадой

problems – описание задач, каждая задача в отдельной папке, содержит входные тесты и ответы на них, а также, возможно, программу-чекер.
report – в данную папку будет сгенерирован отчет.
runs – служебная папка, содержит результаты запусков всех программ участников.
solves – решения участников, каждый участник в отдельной папке.
tmp – папка для временных файлов, используется рабочим каталогом для компиляции и запусков программ.
contest.conf – файл конфигурации олимпиады.

Шаг 1. Создание новой олимпиады

В каталоге *contests* необходимо создать отдельный подкаталог с описанной выше структурой.

Шаг 2. Конфигурирование (файл *contest.conf*)

Данный файл имеет структуру INI-файла, которую можно условно разделить на три части: общие настройки, описание задач с их персональными настройками, и перечень возможных языков программирования.

Секция [contest]

name – название олимпиады.
languages – перечень используемых языков программирования, через запятую перечисляются *названия секций с описаниями ЯП* (подробное описание этих лекция ниже).
problems – количество задач.

problems_filenames_type – тип имен файлов, в которых участники сохраняют решения своих задач, может принимать два значения: *fixed* – файл с решением каждой задачи имеет свое индивидуальное имя (напр., связанное с сюжетом задачи), которое будет указано в секции с описанием задач; *dynamic* – динамическое формирование имен файлов, в данном случае используется следующий параметр - *problems_filenames.*

problems_filenames – имеет значение только в случае установки предыдущего параметра в значение dynamic; описывает формат имен файлов с решениями участников, может содержать шаблоны: %PN% (сокр. от Problem Number) – будет заменено последовательно на числа 1, 2, 3, и т.д., %PL% (сокр. от Problem Letter) – будет заменено последовательно буквами латинского алфавита в верхнем регистре A, B, C, и т.д.

Пример: problems_filenames = "z%PN%"

Подразумевает названия файлов вида z1, z2, z3, и т.д.

Следующие настройки секции [contest] описывают настройки по-умолчанию для всех задач олимпиады, и в последствии, в секции каждой задачи, любой из данных параметров может быть «перекрыт» новым, индивидуальным, значением.

tests – кол-во тестов в задаче.
runs – кол-во запусков программы на каждом тесте, кол-во чисел через запятую должно быть равно кол-ву тестов.
points – баллы за успешное прохождение теста.

input_data_source – источник входных данных для задачи, принимает одно из 4 значений:

- ***none*** – задача не подразумевает входных данных
- ***stdin*** – входные данные программа ожидает на стандартном входном потоке
- ***file*** – входные данные будут прочитаны из файла
- ***auto*** – автоматический режим, является совмещением stdin и file.

input_source_filename – формат имени файла с входными данными теста, имеет значение только в режимах входных данных ***file*** и ***auto***. Может содержать шаблоны %PN% и %PL%.

input_data_filename – формат имен файлов, которые содержат тесты к задаче. Данные файлы должны быть расположены в каталоге problems/имя_задачи/. Помимо шаблонов %PN% и %PL% также используются %TN% и %TL%, которые заменяются на номер теста в числовом или символьном виде соответственно.

output_data_source – источник выходных данных задачи, принимает одно из 3 значений:

- ***stdout*** – задача возвращает результат в стандартный выходной поток
- ***file*** – результат будет записан в файл
- ***auto*** – автоматический режим, проверяются как stdout, так и выходной файл

output_source_filename – формат имени файла с результатом работы программы, имеет значение только в режимах выходных данных ***file*** и ***auto***. Может содержать шаблоны %PN% и %PL%.

answer_data_filename – формат имен файлов, которые содержат правильные результаты на тесты. Данные файлы должны быть расположены в каталоге problems/имя_задачи/. Может содержать шаблоны %PN%, %PL%, %TN% и %TL%.

time_limit – лимит времени на работу программы в пользовательском режиме, в миллисекундах.
memory_limit – лимит памяти на процесс программы, в КБ.

checker – используемый чекер по-умолчанию (подробнее о чекерах ниже).

Секция [tricks]
Не даром имеет именно такое название, потому как она содержит некоторые «уловки и хитрости», связанные с анализом исходного текста программы, и внесением, при необходимости, изменений в этот текст или в параметры проверки.
В данный момент поддерживаются 4 необязательных параметра (значение ***yes*** – включение параметра, ***no*** – отключение):

check_invalid_filenames – проверка правильности имен файлов, используемых программой для ввода и вывода. В случае обнаружения ошибки, вносится изменение и переадресуется на консольный ввод/вывод. Реализован для только языков pascal и delphi.

units_remove – удаление из исходного кода ключевого слова uses и списка всех подключенных модулей. А также удаляется вызов процедуры ClrScr. Реализован только для языка pascal.

void_main_change – если основная функция программы была объявлена как void main(), производится замена на int main() и в конце добавляется return 0. Реализован только для языка C++.

check_random – если в исходном тексте были найдены вызовы функции генерации случайный чисел, и параметр кол-ва запусков для данного теста (*runs*) равен 1, то он будет изменен на 3, для обеспечения точности проверки.

При срабатывании любого параметра из данной секции в примечание к проверке данной задачи добавляется соответствующее предупреждение.

Секции описания задач
Имена секций должны быть вида problemN, где N – это номер задачи, кол-во таких секций должно быть равно общему кол-во задач.

Единственным обязательным параметром этих секций является параметр *name* – имя задачи; одновременно это имя является и именем папки в problems/, которая содержит тесты к задаче.

Если в секции [contest] тип имен файлов с решениями задач (*problems_filenames_type*) было установлено в *fixed*, то вторым обязательным параметром становится *filename*, который содержит уникальное имя файла для данной задачи.

Дополнительными параметрами могут быть любые, из описанных выше в секции [contest].

Секции описания языков программирования
Перечень имен данных секций был указан в параметре *languages* секции [contest].

name – название компилятора.

source_filename – формат имени файла, который должен быть скомпилирован данным компилятором. Содержит в себе шаблон %problemname%, который подставляет имя файла каждой задачи, в динамическом режиме (problems_filenames_type = dynamic) это будут имена, например, z1, z2, z3, и т.д., в случае фиксированных имен, это имя будет взято из параметра *filename* секции описания задачи.

compile_commandline – строка, которая содержит путь к компилятору и параметры, с которыми он должен быть запущен. Может содержать шаблоны:
%mainfile% - имя файла, сформированное из предыдущего параметра *source_filename*
%basename% - то же имя файла, что и в %mainfile%, только без расширения

executable_filename – формат имени файла, который должен быть создан компилятором в случае успешной компиляции. Данное имя файла используется только для проверки, был ли

создан такой файл (компиляция прошла успешно), или нет (ошибка компиляции). Может содержать шаблоны %mainfile% и %basename%.

execution_commandline – командная строка, которая должна быть выполнена для запуска программы на выполнение. Может содержать шаблоны %mainfile% и %basename%.

Единственным необязательным параметром данных секций является *stop_words_file* – путь к файлу, который содержит, так называемые, стоп-слова для данного компилятора (каждое слово должно быть с новой строки в файле). В случае если данный параметр присутствует, и исходный программный код программы содержит хотя бы одно слово из данного списка, то такая программа даже компилироваться не будет, и по данной задаче будет возвращена ошибка безопасности.

Шаг 3. Создание необходимых для проверки файлов

Тесты для задач. В папке problems необходимо создать подкаталоги с тестами и правильными ответами в соответствии с настройками в файле конфигурации.

Решения участников. Необходимо сохранить в папке solves, решения каждого участника должно быть в отдельном подкаталоге.

Шаг 4. Проверка

Когда все готово, можно приступать к проверке решений.
Из командной строки необходимо запустить ts.exe с обязательным ключом -c и указанием папки с олимпиадой, которую необходимо проверить (одна из папок в contests), при отсутствии данного ключа будет выдано соответствующее предупреждение.

Пример: ts.exe -c example

После запуска сразу производятся проверки:

- Всех настроек из contest.conf, в случае наличия ошибок будет выдано сообщение: «Error in the configuration file. Use -tc option for more details.» Ключ -tc пока еще не реализован, поэтому ошибку придется искать самостоятельно.
- Наличие необходимых файлов и папок.

Если все ок – начинается проверка. Результаты проверки будут показаны в консоли в виде:
(для отключения вывода какой-либо информации на консоль, «тихий режим», можно использовать ключ -q)

```
  Competitor User203-12
======================================================================
Problem  Compiler            Results                          Points
======================================================================
Z1       Free Pascal 2.2.2   CE
Z2       Free Pascal 2.2.2   PE PE PE PE PE PE PE TL TL TL  0
Z3       Free Pascal 2.2.2   OK OK OK WA OK OK OK OK OK OK  18
Z4       Free Pascal 2.2.2   OK WA WA WA WA WA WA WA WA WA  2
======================================================================
  Total points:                                               20
======================================================================
```

Поле Results выводит результат проверки по каждому тесту задачи.

Типы ошибок, предусмотренных в проверочной системе:
SE – Security Violation – Ошибка безопасности.
CE – Compilation Error – Ошибка компиляции.

RT – Run-Time Error – Ошибка во время работы программы.
TL – Time Limit Exceeded – Превышен лимит времени.
ML – Memory Limit Exceeded – Превышен лимит памяти.
PE – Presentation Error – Ошибка представления выходных данных (пустой или отсутствует файл с результатом, неверный формат результирующих данных).
WA – Wrong Answer – Неверный ответ.
CF – Check Failed – Ошибка проверочной системы (может быть вызвана самой проверочной системой, подсистемой запуска программ, чекером).
OK – OK.

Дополнительные опции запуска:
-f имя_файла – параллельное дублирование всех данных, выводимых в консоли, в указанный файл.
-u имя_участника[,имя_участника,…] – проверка только указанных участников, их имена перечисляются через запятую.
-p имя_задачи[,имя_задачи,…] – проверка только по указанным задачам, имена задач перечисляются через запятую.
-a – добавление результатов текущей проверки к ранее сохраненным, по-умолчанию, без данного ключа, все предыдущие результаты уничтожаются.

После проверки всех решений:

- Результаты всех запусков программ будут сохранены в каталоге runs.
- «Рядом» с файлом contest.conf будет сформирован файл results.conf, который содержит всю подробную информацию о результатах проверки. Данный файл также является INI-файлом, но его чтение не совсем удобно, поэтому можно переходить к следующему шагу – генерации отчетов.

Шаг 5. Генерация отчетов

На основе всей имеющейся информации: задачи и тесты к ним, решения участников и результаты их проверки, генератор отчетов позволяет сформировать их в удобном для просмотра виде в форме набора HTML-страниц.

Запуск программы Report Builder производится так же, как и проверочная система, из командной строки с ключом -c и указанием папки с олимпиадой.

Пример: rb.exe -c example

После завершения работы данной программы, на основе шаблона отчета в папке report, будет сгенерирован отчет, который будет помещен в папке contests/example/report/.

Внимание! Сгенерированный отчет необходимо скопировать в любой папку WEB-сервера (осторожно, много файлов, при удаленном копировании желательно заархивировать), и просматривать отчет в браузере через протокол HTTP.
Данная необходимость в веб-сервере вызвана тем, что приложение с отчетом использует технологию AJAX для динамической подгрузки необходимых файлов, а данная технология работает только через веб-сервер. Размещение всего объема сгенерированной информации сразу внутри одной страницы в результате тестирования было признано нерациональным в связи с необходимостью большого кол-ва оперативной памяти для работы браузера.

Описание системы отчетов пока не прилагаю, надеюсь на интуитивно-понятный интерфейс J Двойной клик на любом участнике открывает окно с его результатами, двойной клик на

любой задаче в этом окне открывает подробную информацию прохождения выбранного теста.

Чекеры

Для проверки результата работы программы используются программы-чекеры. Они могут быть встроенными и внешними.

Встроенные чекеры
Если в качестве имени чекера в задаче указано ключевое слово *internal:* то будет использован один из встроенных чекеров, указанный после двоеточия.

Возможные варианты:
strings – построчное сравнение результата с правильным ответом, пробелы в начале и в конце строк опускаются.
strings+spaces – то же, что strings, только с учетом начальных и конечных пробелов.
strings-spaces – то же, что strings, только пробелы опускаются не только в начале и в конце, но и между слов (лишние пробелы, 1 остается).
int – сравнение целых 32-битных чисел со знаком.
double – сравнение 64-битных чисел с плавающей запятой (сравнивать дробные числа конечно не есть хорошо, этот момент я еще пересмотрю).

Внешние чекеры
В качестве параметра ***checker*** в файле конфигурации может быть указано имя программы-чекера. Данная программа должна располагаться в папке с тестами к задаче.
Чекер запускается с 3 параметрами:

checker.exe [путь к файлу с входным тестом] [путь к файлу с результатом участника] [путь к файлу с правильным ответом]

Программа должна вернуть в выходной поток результат проверки в виде XML-пакета следующего формата:

```
<?xml version="1.0"?>
<check status="OK">
	<comment>OK</comment>
</check>
```

Атрибут status содержит результат проверки. Он должен принимать одно из 4 состояний:

- WA – Ответ неправильный.
- PE – Неверный формат результата.
- CF – Ошибка чекера.
- OK – Правильный ответ.

Тег <comment> может содержать дополнительную информацию, которую потом можно будет посмотреть в отчете проверки.

Данная структура ответа обязательна для успешной работы чекера. Символы табуляции и переноса строк здесь показаны для наглядности, в ответе чекера их может и не быть.