Министерство транспорта Российской федерации

Федеральное агентство железнодорожного транспорта

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Дальневосточный государственный университет путей сообщения»

Кафедра «Вычислительная техника и компьютерная графика»

К защите допустить

Заведующий кафедрой

\_\_\_\_\_ Ю.В. Пономарчук

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2016

Система для проведения соревнований по программированию с модулем подведения итогов

Выпускная квалификационная работа

ВКР 09.03.01.ИВТ.17.00.943 – ПЗ

Студент 943 гр. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Усманов

Руководитель

к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.А. Сухобок

Нормоконтролёр

к.т.н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Буняева

Хабаровск 2016

Содержание

[Введение 3](#_Toc454055240)

[1 Обзор предметной области 5](#_Toc454055241)

[1.1 Соревнования по программированию 5](#_Toc454055242)

[1.1.1 Командный чемпионат по программированию ACM ICPC 5](#_Toc454055243)

[1.1.2 Правила командных чемпионатов 6](#_Toc454055244)

[1.1.3 Подведение итогов соревнований по программированию 7](#_Toc454055245)

[2 Проектирование системы для проведения соревнований по программированию с модулем подведения итогов 8](#_Toc454055252)

[2.3 Требования, предъявляемые к модулю подведения итогов 8](#_Toc454055263)

[3 Реализация системы для проведения соревнований по программированию с модулем подведения итогов 9](#_Toc454055264)

[3.2 Разработка модуля подведения итогов 9](#_Toc454055271)

[3.2.1 Реализация меню настроек 9](#_Toc454055272)

[3.2.2 Хранение историй посылок на сервере 10](#_Toc454055273)

[3.2.3 Реализация загрузки истории посылок 11](#_Toc454055274)

[3.2.4 Формат историй посылок 14](#_Toc454055275)

[3.2.5 Обработка историй посылок 17](#_Toc454055276)

[3.2.6 Хранение информации о командах, задачах и посылках 19](#_Toc454055277)

[3.2.7 Структура таблицы с результатами 21](#_Toc454055278)

[3.2.8 Цветовая схема таблицы с результатами 23](#_Toc454055279)

[3.2.9 Обновление таблицы с результатами 25](#_Toc454055280)

[3.2.10 Реализация «разморозки» таблицы с результатами 26](#_Toc454055281)

[3.3 Технико-экономическое обоснование 29](#_Toc454055282)

[Заключение 31](#_Toc454055283)

[Список использованных источников 32](#_Toc454055284)

Введение

Олимпиады по программированию в век наступления «промышленного программирования» остаются одним из немногих интеллектуальных конкурсов, на которых участники могут продемонстрировать свои способности в искусстве программирования в его классическом понимании и которые поддерживают традиции этой области теоретической информатики.

На первый взгляд кажется, что эти состязания относятся к области «высокого чистого искусства», имеющего мало точек соприкосновения с «реальной жизнью» современного программиста. Действительно, круг из нескольких десятков одаренных молодых людей, реально претендующих на победу в мировом первенстве, и их тренеров весьма узок, математико-программистское содержание состязаний понятно достаточно ограниченному числу специалистов и педагогов высшей квалификации.

Однако, результаты этих интеллектуальных игр мировой программисткой элиты представляет интерес уже для гораздо более широких кругов, поскольку процесс формирования и подготовки команд опирается на всю образовательную, научную, промышленную и культурную инфраструктуру данной страны. Эти результаты достаточно точно отражают распределение мирового компьютерного интеллектуального потенциала по различным странам и позволяют оценить способность нации не только использовать разработанные в других странах компьютерные технологии, но и вносить свой вклад в их создание.

Проведение соревнований по программированию играет немалую роль при подготовке специалистов сферы информационных технологий. Подведение итогов подобных соревнований происходит по определенным правилам, и, зачастую, на награждении используется стороннее от системы проведения соревнований программное обеспечение.

Целью выпускной квалификационной работы является проектирование и разработка системы для проведения соревнований по программированию с модулем подведения итогов.

В соответствии с указанной целью поставлены следующие задачи:

– изучить предметную область: правила проведения и подведения итогов соревнований по программированию;

– произвести анализ существующих систем для проведения соревнований по программированию;

– произвести анализ существующих модулей подведения итогов, анализ способов и средств их реализации;

– выполнить проектирование системы для проведения соревнований по программированию с модулем подведения итогов;

– разработать спроектированную систему.

Разработанная система будет внедрена и начнёт использоваться в подготовке студентов, проходящих обучение на кафедре «Вычислительная техника и компьютерная графика» «Дальневосточного государственного университета путей сообщения».

1 Обзор предметной области

Системы для проведения соревнований по программированию могут автоматизировать процесс подготовки студентов к самим соревнованиям по программированию, а так же развить их навыки программирования в целом.

1.1 Соревнования по программированию

Можно выделить следующие регулярные мировые соревнования по программированию [1]:

– командный чемпионат ACM ICPC, проводимый компанией IBM [2];

– командный чемпионат VK Cup [3];

– чемпионат Facebook Hacker Cup [4];

– чемпионат Google Code Jam [5];

– чемпионат Russian Code Cup, проводимый компанией Mail.ru Group [6];

– чемпионат Алгоритм, проводимый компанией Яндекс [7].

Самым престижным и развитым является командный чемпионат ACM ICPC, в то время как остальные мировые соревнования проводятся по большей части в онлайн режиме.

1.1.1 Командный чемпионат по программированию ACM ICPC

Командные чемпионаты мира по программированию (International Collegiate Programming Contest) среди сборных команд высших учебных заведений проводятся с 1977 г. под эгидой международной организацией Association for Computing Machinery (ACM), которая была основана в 1947 г. сотрудниками Пенсильванского университета, создавшим первый в мире компьютер ENIAC [2].

Схема организации чемпионата включает два этапа: региональный и финальный. Команды-победительницы региональных групп выходят в финал и разыгрывают звания чемпионов мира, Европы, Азии и других континентов.

Впервые о чемпионате мира российские вузы узнали осенью 1996 г., когда был образован новый Восточно-Европейский регион. За более чем двадцатилетнюю историю полуфинала чемпионата мира в России, российские команды достойно выступали в финалах. Например, российские команды Санкт-Петербургского государственного университета информационных технологий, механики и оптики занимали первое место в финалах в 2004, 2008, 2009, 2012, 2013, 2015 годах.

1.1.2 Правила командных чемпионатов

За почти сорокалетнюю историю состязаний сформировались международные правила [8] этих соревнований, в соответствии с которыми команде, состоящей из трех участников, предоставляется один компьютер и предлагается в течение 5 часов решить максимальное число из предложенных задач. Количество задач обычно варьируется от 8 до 15. Побеждает команда, решившая наибольшее число задач, а в случае равенства числа решенных задач – команда, затратившая меньше времени.

Время, затраченное на решение задач – это время в минутах, прошедшее с начала соревнований до посылки первой принятой попытки для этой задачи плюс заданное количество штрафных минут за каждую неудачную попытку для этой задачи. Время, затраченное на нерешенную задачу, равно нулю. Общее время равно сумме времен, затраченных на решение всех задач.

Решением является программа на одном из предложенных языков программирования. Тестирование производится автоматически по ходу соревнований в специальной тестирующей системе на заранее подготовленном наборе тестов для каждой задачи. Решения задач, посланные жюри, называются попытками. Каждая попытка может быть принята или отвергнута. Попытка считается принятой, если программа выдает верные результаты и укладывается в ограничения на всех тестах.

В различных соревнованиях правила могут немного отличаться от описанных. Например, в соревнованиях Google Code Jam и Facebook Hacker Cup тестирование программ происходит на компьютере участника, после чего он отсылает результаты и свою программу жюри.

1.1.3 Подведение итогов соревнований по программированию

Системы для проведения соревнований по программированию позволяют получать результаты тестирования решений участников в течение нескольких минут после их отправления. Все участники и зрители в реальном времени могут наблюдать за тем, какое количество задач решила каждая команда посредством таблицы результатов. Такой формат проведения соревнований исключает какую-либо интригу относительно победителя, так как он известен сразу по окончании соревнования.

Для сохранения интриги, в соответствии с правилами, таблица результатов замораживается за час до конца соревнования [9]. Так называемая «разморозка» таблицы происходит лишь на церемонии награждения по определённому алгоритму. Вместо того чтобы просто показать участникам финальную таблицу, её «размораживают» последовательно снизу вверх. В случае, если команда в последний час имела успешные посылки, она поднимается вверх по таблице, что увеличивает её шансы на победу. Иначе, если удачных посылок у команды не осталось – она приглашается на сцену для награждения.

В настоящее время существует множество модулей, позволяющих «разморозить» таблицу результатов, но все они функционируют отдельно от систем для проведения соревнований. Такие модули не имеют открытого распространения, поэтому сложно рассматривать способы их реализации. Часто можно встретить два способа реализации модуля подведения итогов: в виде web-страницы и в виде исполняемого файла. Первый вид реализации является более универсальным, так как может быть запущен с различных платформ, в отличие от исполняемого файла, который, скорее всего, будет привязан к конкретной операционной системе. Более того, web-страница не потребует установки, а будет иметь открытый доступ через интернет.

Исходя из этого, следует рассмотреть средства и технологии для разработки web-сайтов и определиться с требованиями, предъявляемыми к модулю подведения итогов.

2 Проектирование системы для проведения соревнований по программированию с модулем подведения итогов

2.3 Требования, предъявляемые к модулю подведения итогов

Исходя из рассмотренных ранее технологий для создания web-сайтов, отображать модуль подведения итогов можно посредством одной web-страницы, содержащей таблицу с результатами и меню настроек.

Помимо отображения таблицы с результатами необходимо добиться так называемой «разморозки» этой таблицы, то есть последовательного заполнения таблицы результатами в том порядке, в котором они получались по ходу соревнования.

Меню настроек должно включать в себя следующий перечень функций:

– выбор формата истории посылок;

– выбор момента «заморозки» таблицы;

– обнуление таблицы и переход на момент «заморозки»;

– регулирование скорости анимации «разморозки»;

– последовательная или хронологическая «разморозка». Под хронологической «разморозкой» подразумевается заполнение таблицы результатами в хронологическом порядке. Алгоритм последовательной «разморозки» начинает обработку посылок самого нижнего участника в таблице, у которого еще остались «замороженные» посылки. В случае, если его посылка была успешной – он поднимается вверх по таблице;

– пошаговая или автоматическая «разморозка»: то есть «разморозка» по нажатию кнопки или же без него соответственно;

– включение и выключение отображения времени посылок;

– включение и выключение отображения будущих посылок;

– обработка только последней посылки.

История посылок с результатами соревнований должны загружаться с сервера или с компьютера пользователя.

3 Реализация системы для проведения соревнований по программированию с модулем подведения итогов

3.2 Разработка модуля подведения итогов

По требованиям, предъявляемым к модулю подведения итогов, он может быть реализован посредством одной web-страницы. На данной странице размещены меню настроек и таблица с результатами. Помимо этого присутствуют две кнопки – первая для вызова меню, вторая для управления «разморозкой».

3.2.1 Реализация меню настроек

Всплывающее меню с настройками (рисунок 3.1) описанными в требованиях реализовано с помощью HTML, CSS, JavaScript и библиотеки jQuery.

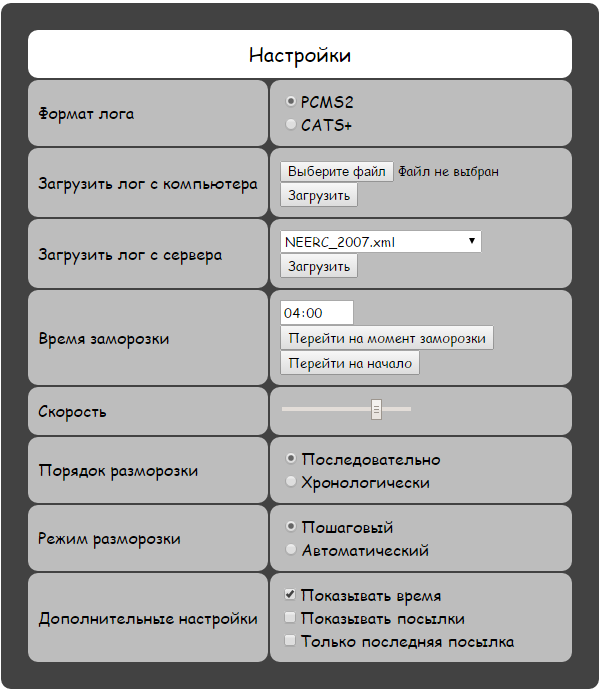


Рисунок 3.1 – Меню с настройками «разморозки»

Меню с настройками имеет так называемую табличную вёрстку страницы, которая определена тэгом table в файле index.php (приложение А, листинг А.1).

Оформление меню определено свойствами тэгов в файле setting.css (приложение А, листинг А.2).

Выпадающий список в меню содержит в себе информацию о том, какие истории посылок имеются на сервере. Формирование данного списка будет рассмотрено в следующем подразделе.

Вызов меню осуществляется посредству одной из кнопок на web-странице. При нажатии на кнопку меню плавно исчезнет или появится. Исчезновение сопровождается изменениями размера меню и параметра прозрачности, благодаря чему исчезновение выглядит так, как будто меню сжимается в одну точку. Такая анимация реализована с помощью библиотеки jQuery (листинг 3.1).

Листинг 3.1 – Реализация анимации исчезновения/появления меню

var showSetting = true;

function hideSetting() {

if( showSetting )

$( "#setting" ).hide( "slow" );

else

$( "#setting" ).show( "slow" );

showSetting = !showSetting;

}

Функции hide() и show() библиотеки jQuery позволяют добиться описанной выше анимации.

3.2.2 Хранение историй посылок на сервере

На сервере в папке log хранятся файлы формата XML с историями посылок различных соревнований. Точно такой же формат имеют истории посылок, которые экспортируются из системы Ejudge.

Помимо этого на сервере имеется база данных с наименованиями всех имеющихся историй посылок. Подключение к базе данных осуществляется через PHP-сценарий db.php (листинг 3.2).

Листинг 3.2 – Файл db.php

<?php

$dbhost = 'localhost'; $dbname = 'table';

$dbuser = 'root'; $dbpass = '';

$connection = mysql\_connect( $dbhost, $dbuser, $dbpass ) or die( mysql\_error() );

mysql\_select\_db( $dbname, $connection ) or die( mysql\_error() );

?>

Наименования историй посылок хранятся в базе данных для быстрого формирования выпадающего списка из меню настроек, которое реализовано вызовом PHP-сценария logs.php (листинг 3.3):

<?php include "php/logs.php"; ?>

Листинг 3.3 – Файл logs.php

<?php

include "php/db.php";

$query = "SELECT \* FROM `logs`";

$res = mysql\_query( $query );

echo "<select id = 'fileLogFromServer'>";

while( $row = mysql\_fetch\_array( $res ) ) {

echo "<option>".$row["name"]."</option>";

}

echo "</select>";

?>

Сценарий logs.php делает запрос в базу данных для получения списка наименований историй посылок. После этого, через цикл, идёт заполнения тэга select, который в свою очередь является выпадающим списком. Элементы выпадающего списка выделяются тэгами option.

3.2.3 Реализация загрузки истории посылок

Как описано ранее, реализовано два варианта загрузки истории посылок:

– с компьютера пользователя;

– с сервера.

Для загрузки истории посылок с компьютера используется стандартный элемент оформления web-страниц – элемент input с типом file. После выбора файла (рисунок 3.2), он обрабатывается через JavaScript (листинг 3.4).

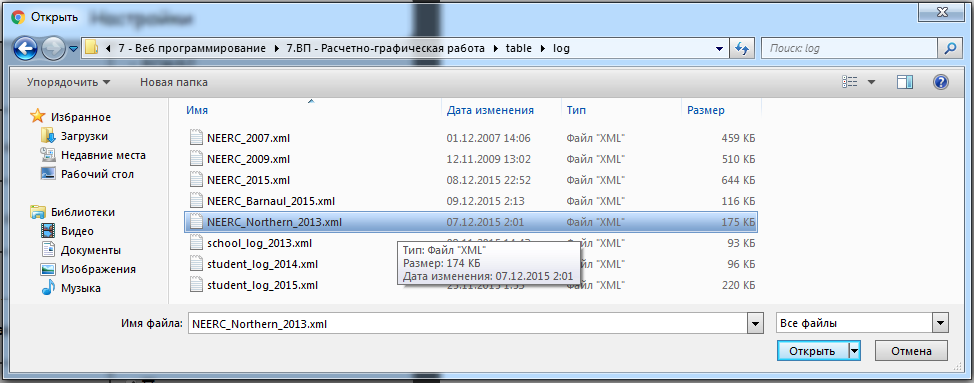


Рисунок 3.2 – Выбор истории посылок для загрузки

Листинг 3.4 – Обработка загружаемого файла

function loadLogFromUser() {

if( loadLog( function( log ) {

isLoadLog = false;

loadLogFromFile( log, teams, problems, submits );

createTable();

goStart();

} ) ) {

deleteTable();

}

}

function loadLog( onSuccess ) {

var files = document.getElementById( "fileLog" ).files;

if( !files.length ) {

return false;

}

var reader = new FileReader();

reader.onloadend = function( event ) {

onSuccess( event.target.result );

};

reader.onerror = function( event ) {

alert( "Файл не может быть прочитан! Код ошибки: " + event.target.error.code );

};

isLoadLog = true;

reader.readAsText( files[0] );

return true;

}

Функция loadLogFromUser() вызывается при загрузке истории посылок с компьютера. Данная функция с функцией onSuccess() в качестве аргумента, вызывает функцию loadLog(), которая в свою очередь, пытается считать файл с историей посылок. В случае успеха, функция loadLog() вызывает переданную функцию onSuccess(), которая отвечает за получение необходимых данных из истории посылок.

Выбор истории посылок для загрузки с сервера осуществляется через выпадающий список (рисунок 3.3).

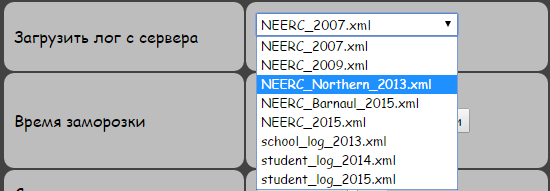


Рисунок 3.3 – Выбор истории посылок для загрузки с сервера

Загрузка истории посылок с сервера реализована с помощью технологии AJAX, которая позволяет функциям JavaScript вызывать PHP-сценарии на стороне сервера и обрабатывать данные, полученные в ответ (листинг 3.5).

Листинг 3.5 ­– Загрузка истории посылок с сервера

function loadLogFromServer() {

deleteTable();

isLoadLog = true;

var file = document.getElementById( "fileLogFromServer" ).value;

$.get( "php/loadLog.php", {"file": file}, function(data) {

isLoadLog = false;

loadLogFromFile( data, teams, problems, submits );

createTable();

goStart();

} );

}

Функция loadLogFromServer() вызывает функцию get() и передаёт в неё функцию onSuccess(), которая рассматривалась ранее. Функция get() делает запрос на сервер с вызовом PHP-сценария loadLog.php (листинг 3.6).

Листинг 3.6 – Файл loadLog.php

<?php

$file = $\_GET["file"];

echo file\_get\_contents( "C:/openServer/domains/table/log/".$file );

?>

PHP-сценарий loadLog.php считывает историю посылок из определённого файла, хранимого на сервере, и возвращает её в функцию onSuccess() для дальнейшей обработки.

Алгоритм обработки истории посылок будет рассмотрен в последующих подразделах.

3.2.4 Формат историй посылок

Истории посылок соревнований могут быть записаны в обычный текстовый файл. Но чаще всего для этого используются файлы формата XML (англ. eXtensible Markup Language – расширяемый язык разметки).

Как описано ранее, в системе Ejudge имеется возможность экспорта истории посылок в формате PCMS2. Именно такой формат реализован в модуле подведения итогов. Помимо этого, так же реализован и формат CATS+.

Формат CATS+

Формат CATS+ является расширенным по отношению к формату CATS. Формат CATS используется Дальневосточным Федеральным Университетом в местных соревнованиях по программированию.

Основным отличием формата CATS+ от CATS является наличие двух дополнительных блоков: с информацией по участникам соревнования и с информацией по задачам.

Истории посылок формата CATS+ представляют собой древообразную модель из тэгов (листинги 3.7-3.9), внутри которых заключена информация о соревновании.

Листинг 3.7 – Блок teams

<teams>

<team>

<id> Идентификатор команды </id>

<name> Название команды </name>

</team>

...

</teams>

Информация о каждой команде помещается в блок team. В качестве идентификатора команды должно выступать число. Название команды может быть любой строкой символов.

Листинг 3.8 – Блок problems

<problems>

<problem>

<code> Сокращенное обозначение задачи </code>

</problem>

</problems>

Информация о каждой задаче помещается в блок problem. В качестве сокращенного обозначения задачи может выступать как число, так и буква, или даже строка символов. Разумеется, лучше всего, если обозначение будет состоять из одного символа, чтобы в таблице c результатами оно не занимало много места, и не привело к увеличению ширины столбцов.

Листинг 3.9 – Блок reqs

<reqs>

<req>

<time\_since\_start> Время посылки </time\_since\_start>

<team\_id> Идентификатор команды </team\_id>

<code> Сокращенное обозначение задачи </code>

<state> Статус посылки </state>

</req>

...

</reqs>

Информация о каждой посылке помещается в блок req. Время посылки – время от начала соревнования в сутках. Идентификатор команды должен быть таким же, как в блоке teams. Сокращенное обозначение задачи должно быть таким же, как в блоке problems. Статус посылки может иметь различные значения, но за удачную посылку будет приниматься лишь статус «accepted».

Посылки из блока reqs с командами или задачами, которые отсутствуют в блоках teams и problems соответственно, игнорируются.

Стоит отметить, что в формате CATS+ в блоке reqs могут присутствовать и иные параметры, которые в текущей реализации модуля подведения итогов не используются.

Формат PCMS2

Формат PCMS2 используется в некоторых соревнования по программированию, проводимых на территории Российской федерации. Примером такого соревнования является NEERC ACM ICPC.

Формат PCMS2, так же как и формат CATS+, имеет древообразную модель из тэгов, но информация записывается не внутри тэгов, а в качестве параметров тэга.

Основным блоком является блок standings, внутри которого расположен блок contest (листинг 3.10).

Листинг 3.10 – Блоки standings и contest

<standings>

<contest name = "Название соревнования">

...

</contest>

</standings>

Внутри блока contest располагается один блок challenge (листинг 3.11) и несколько блоков session (листинг 3.12).

Листинг 3.11 – Блок challenge

<challenge>

<problem alias = "Сокращённое обозначение задачи" />

...

</challenge>

Блок challenge содержит в себе информацию о задачах соревнования. Информация о каждой задаче помещается в блок problem. Сокращенное обозначение задачи будет отображаться в таблице с результатами.

Листинг 3.12 – Блок session

<session party = "Название команды">

<problem>

...

</problem>

...

</session>

Блок session содержит в себе информацию только о посылках конкретного участника, причем все посылки распределены по задачам и заключены в блоки problem (листинг 3.13).

Листинг 3.13 – Блок problem

<problem alias = "Сокращенное обозначение задачи">

<run accepted = "Статус посылки" time = "Время" />

...

</problem>

Блок problem содержит в себе информацию о посылках по конкретной задаче. Сокращенное обозначение задачи должно быть таким же, как и в блоке challenge.

Информация о посылке заключена в блок run и имеет два параметра: статус посылки и время. Статус посылки может иметь два значения: «yes» или «no», первый из которых обозначает, что задача решена. Время – время в миллисекундах от начала соревнования до момента прихода посылки.

Стоит отметить, что в формате PCMS2 могут присутствовать и иные параметры, которые в текущей реализации модуля подведения итогов не используются.

3.2.5 Обработка историй посылок

Как описано ранее, истории посылок хранятся в файлах формата XML. Существует множество автоматизированных способов обработки файлов данного формата. Но для модуля подведения итогов написан свой собственный обработчик.

Для обработки файлов формата XML написан класс StringWithIndex (приложение А, листинг А.4), который имеет два поля – text и index. В поле text помещается содержимое файла XML, а в поле index – текущее положение курсора. Рассмотрим основные методы данного класса:

– методы getNextTagValue() и getTagValue() принимают в качестве аргумента наименование тэга, выполняют поиск этого тэга в поле text правее положения курсора index и возвращают содержимое этого тэга. Разница между методами заключается в том, что первый перемещает положение курсора index в конец найденного тэга;

– методы getNextTagText() и getTagText() принимают в качестве аргумента наименование тэга, а после вызывают методы getNextTagValue() и getTagValue() соответственно. Разница от работы последних заключается в том, что методы getNextTagText() и getTagText() возвращают не сам текст, а объект класса StringWithIndex;

– метод nextTagText() возвращает содержимое ближайшего к курсору тэга в виде объекта класса StringWithIndex, курсор при этом перемещается в конец найденного тэга;

– метод getTagName() возвращает наименование тэга, но поиск выполняется с начала поля text, курсор при этом не перемещается;

– метод getParamsValue() принимает в качестве аргумента наименование параметра, поиск которого выполняется внутри поля text. Если поиск успешен, то метод возвращает значение этого параметра.

Так как модуль подведения итогов поддерживает обработку историй посылок двух разных форматов CATS+ и PCMS2, реализовано два набора функций для работы с каждым из форматов (приложение А, листинги А.5-А.6).

Функции для обработки имеют одинаковые названия – loadLogFromFile(), которые в качестве аргументов принимают на вход содержимое файла XML, а также три массива teams, problems и submits.

Переключение между функциями для обработки для разных форматов осуществляется через меню с настройками (рисунок 3.1). Если произведено изменение формата с сервера загружается файл .js с необходимыми для обработки функциями (листинг 3.14).

Листинг 3.14 – Загрузка файла .js с функциями для обработки

function setScriptLoad( scriptName ) {

var div = document.getElementById( "scriptLoad" );

div.removeChild( div.firstChild );

var script = document.createElement( "script" );

script.setAttribute( "type", "text/javascript" );

script.setAttribute( "src", scriptName );

div.appendChild( script );

}

Функция loadLogFromFile() вызывает три другие функции – getTeamsFromFile(), getProblemsFromFile(), getSubmitsFromFile(), которые, в свою очередь, осуществляют обработку истории посылок и помещают необходимую информацию о командах, задачах и посылках в массивы teams, problems и submits соответственно. Структура данных массивов будет рассмотрена в следующем подразделе.

3.2.6 Хранение информации о командах, задачах и посылках

Как описано ранее, для хранения информации о командах, задачах и посылках в модуле подведения итогов используется три массива teams, problems и submits. Для каждого из массивов написан свой класс с необходимыми полями и методами. Стоит заранее отметить, что примеры использования методов связанных с информацией, отображаемой в таблице с результатами, будут приведены в следующих подразделах.

Класс Team

Класс Team (приложение А, листинг А.7) используется для хранения информации о командах и имеет следующие поля:

– поле id – идентификатор команды;

– поле name – название команды;

– массивы problemAllSubmit и problemSubmit – общее количество посылок и количество обработанных посылок по каждой из задач соответственно;

– массивы problemSolved и problemTime – для хранения информации о том, решена ли конкретная задача и, если решена, то за какое время;

– двумерный массив problemSubmits – для хранения списка идентификаторов посылок, совершенных командой, по каждой из задач;

– поле solved – количество решенных задач;

– поле time – суммарное время, истраченное на решение задач;

– массив submit – для хранения списка идентификаторов всех посылок, совершенных командой;

– поле nextSubmit – номер следующей посылки, которую необходимо обработать.

Данный класс содержит следующие методы:

– методы getSolved() и getTime() возвращают в виде текста количество решенных задач и затраченное время соответственно;

– метод getVerdict() собирает информацию о конкретной задаче для отображения в ячейке таблицы с результатами;

– метод getClassVerdict() определяет класс CSS для ячейки таблицы с результатами по конкретной задаче;

– метод getTimeProblem() переводит время из общего формата (количество минут) к привычному (чч:мм) для конкретной задачи;

– методы addSubmit(), addOneSubmit() и addAllSubmit() используются для обработки посылок: первые два метода обрабатывают конкретную посылку, а третий метод обрабатывает все посылки по конкретной задаче;

– метод addProblemSubmit() добавляет информацию о конкретной посылке по какой-нибудь задаче в двумерный массив problemsSubmits;

– метод solvedProblem() отмечает конкретную задачу решенной и вычисляет общее штрафное время;

– метод clearProblemSubmits() отчищает двумерный массив problemSubmits от данных;

– метод createProblems() инициализирует все поля объекта связанные с задачами, то есть, например, problemAllSubmit, problemSubmit, solved и т.д.;

– метод getIdSubmit() возвращает идентификатор следующей посылки, которую необходимо обработать;

– методы getIdPlace(), getIdName(), getIdSolved(), getIdTime() и getIdProblem() используются для генерации идентификатора ячейки таблицы с результатами.

Класс Problem

Класс Problem (приложение А, листинг А.8) используется для хранения информации о задачах и имеет следующие поля:

– поле code – идентификатор задачи;

– поле solved – количество команд, решивших данную задачу;

– поле submit – количество попыток решить данную задачу.

Данный класс содержит следующие методы:

– метод getSolved() собирает информацию по задаче для отображения в ячейке таблицы с результатами;

– метод getClassSolved() определяет класс CSS для ячейки таблицы с результатами по задаче;

– метод addSubmit() обрабатывает конкретную посылку, обновляя данные о количестве посылок по задаче;

– метод getIdStatistic() используется для генерации идентификатора ячейки таблицы с результатами.

Класс Submit

Класс Submit (приложение А, листинг А.9) используется для хранения информации о посылках и имеет следующие поля:

– поле time – время прихода посылки;

– поле teamId – идентификатор команды, которая отправила посылку;

– поле problemCode – идентификатор задачи, по которой совершена посылка;

– поле state – статус посылки, значение «accepted» означает, что задача решена;

– поле test – номер теста, на котором решение выдало неверный ответ;

– поле checked – флаг, отмечающий обработана посылка или нет.

Данный класс содержит только один метод getTime(), который используется для вычисления округлённого времени прихода конкретной посылки в минутах.

3.2.7 Структура таблицы с результатами

Таблица с результатами состоит из следующих элементов:

– заголовок таблицы;

– основная часть таблицы с результатами участников.

Заголовок таблицы содержит три строки:

– строка с названиями столбцов;

– строка со статистикой соревнования;

– разделитель между заголовком и остальной таблицей.

Столбцы таблицы:

– место участника. Стоит отметить, что если два участника решили одинаковое количество задач с одинаковым штрафным временем, то они занимают одно место;

– название команды;

– количество решенных задач;

– штрафное время;

– а так же по столбцу на каждую задачу.

Стоит отметить, что в заголовке в строке статистика отображается количество удачных посылок и количество посылок по каждой задаче, а так же общее количество удачных посылок и общее количество посылок по всем задачам.

Формирование таблицы происходит после загрузки истории посылок (приложение А, листинг А.10) и представляет собой генерацию HTML кода web-страницы. Для того, чтобы данная генерация занимала минимальное время работы, она происходит на стороне клиента посредству JavaScript, а не на стороне сервера.

Для быстрого доступа к конкретным ячейкам таблицы в дальнейшем каждой из них присваивается уникальный идентификатор. Как описано ранее, классы Team и Problem содержат методы для генерации данных идентификаторов. Примеры таких идентификаторов: «ID команды.Name», «ID команды.Place», «ID задачи.statistic» и т.д.

Помимо генерации таблицы в файле createTable.js присутствуют функции для заполнения дополнительных переменных информацией о командах, задачах и посылках. Например, функция createSubmits() использует компаратор compareSortSubmit() для сортировки посылок по времени их отправления участниками.

Определив структуру таблицы с результатами, стоит рассмотреть её дизайн.

3.2.8 Цветовая схема таблицы с результатами

Цветовая схема таблицы – очень важная составляющая общего дизайна модуля подведения итогов, так как от этого зависит понимание пользователем происходящего. Весь дизайн таблицы с результатами задаётся свойствами классов CSS (приложение А, листинг А.3).

Заголовок таблицы имеет черный цвет, за исключением ячеек с количеством посылок по конкретной задаче (рисунок 3.4).

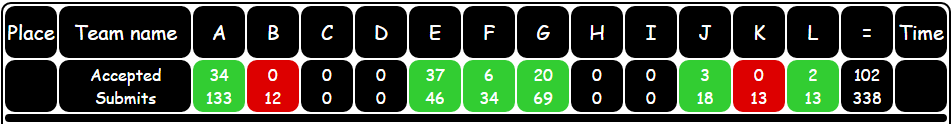


Рисунок 3.4 – Заголовок таблицы результатов

Строки с участниками имеют два оттенка серого цвета (рисунок 3.5). Два участника будут иметь один и тот же цвет, если у них решено одинаковое количество задач. Два участника будут иметь разный цвет, если нет участника, который решил больше задач, чем один, и меньше задач, чем другой, из них.

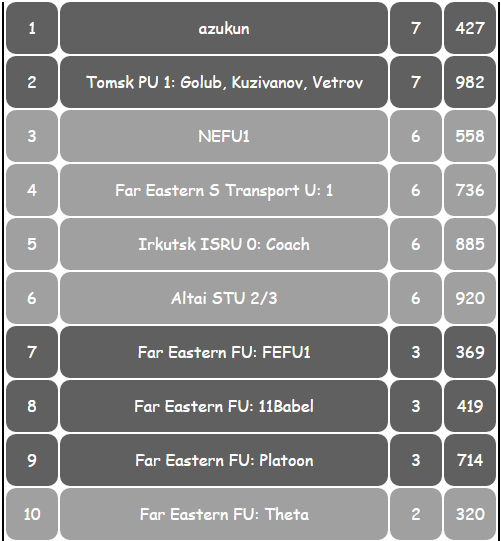


Рисунок 3.5 – Цветовое выделение участников

Таким образом, участники делятся на блоки, объединённые одним цветом.

Цвета для центральной части таблицы представлены на рисунке 3.6.



Рисунок 3.6 – Цвета центральной части таблицы

Каждый из этих цветов используется для своей цели:

– зелёный, если участник решил задача;

– тёмно-зелёный, если участник первым решил задачу;

– красный, если участник сделал неудачные посылки по задаче;

– синий, если участник сделал посылки, результат которых еще не отображен в таблице. Например, если он делал их в период «заморозки» таблицы результатов;

– мигающий сине-оранжевый для выделения посылки, которая подвергнется «разморозке»;

– бледно серый, если участник не делал посылок по задаче.

Стоит отметить, что при последовательной «разморозке» участник, посылки которого рассматриваются, так же выделяется оранжевым цветом (рисунок 3.7), для того чтобы сфокусировать внимание пользователя.



Рисунок 3.7 – Выделение участника при последовательной «разморозке»

Некоторые из цветов, представленных на рисунке 3.6, используются в заголовке таблицы результатов. Зелёный цвет используется для задач, по которым есть хотя бы одна удачная посылка, в то время как красный цвет для задач, по которым есть хотя бы одна неудачная посылка (рисунок 3.8).



Рисунок 3.8 – Часть заголовка таблицы результатов с информацией по задачам

Для задач, по которым нет посылок, цвет остаётся черным.

3.2.9 Обновление таблицы с результатами

Для обновления таблицы с результатами во время «разморозил» используется множество функций (приложение А, листинг А.11). Основные из них:

– функция updateTable() полностью обновляет таблицу с результатами;

– функция sortTr() перемещает определённую строку таблицы на определённую позицию;

– функция sortTable() перемещает все строки таблицы по местам, на которых они должны находиться в текущий момент;

– функция sortTrSubmit() запускает анимацию плавного перемещения строки в таблице;

– функции sortTrNextSubmit() и sortTrNextTeamSubmit() определяют положение команды в таблице и запускают анимацию сортировки таблицы;

– функция updateTableClass() обновляет классы CSS для всех ячеек таблицы;

– функция updateTrClass() обновляет классы CSS для всех ячеек одной строки;

– функции updateTdPlaceClass(), updateTdNameClass(), updateSolvedClass() и updateTdTimeClass() отвечают за обновление классов CSS ячеек с местом, названием команды, количеством решенных задач и штрафным временем соответственно;

– функции updateTrPlace() и updateTdPlace() отвечают за обновление мест для всех строк таблицы и обновления места для одной строки соответственно;

– функция updateTrSolvedAnsTime() обновляет количество решенных задач и штрафное время для всех команд;

– функции onTdNextSubmit(), ontTdNextTeamSubmit(), offTdNextSubmit() и offTdNextTeamSubmit() определяют команду и задачу по обрабатываемой в текущий момент посылке;

– функции onTdSubmit() и offTdSubmit() отвечают за включение и выключение анимации подсветки ячейки таблицы;

– функции onTdTeam() и offTdTeam() отвечают за включение и отключение подсветки названия команды при последовательной «разморозке»;

– функции updateTrStatistic() и updateTdStatisticProblem() отвечают за обновление всей строки статистики и одной ячейки в строке статистики в заголовке таблицы;

– функция updateTdTotal() обновляет ячейку с общим количество удачных и неудачных посылок;

– функция getTotal() генерирует HTML код с информацией об общем количестве удачных и неудачных посылок для отображения в соответствующей ячейке таблицы с результатами;

– функции updateTdSolvedAndTime(), updateTdSolved() и updateTdTime() отвечают за обновление количества решенных задач и обновление штрафного времени для конкретной команды;

– функция updateTableProblem() обновляет ту часть таблицы, которая связана с посылками команд по задачам;

– функция updateTdProblem() обновляет одну ячейку, которая связана с посылкой конкретной команды по конкретной задаче.

Стоит отметить, что для обновления некоторой информации в ячейках таблицы с результатами, функции используют методы классов Team и Problem, отвечающие за генерацию соответствующей информации. В частности, используются функции для генерации идентификаторов ячеек таблицы.

3.2.10 Реализация «разморозки» таблицы с результатами

После того, как история посылок загружена и обработана, сгенерирована таблица с результатами, можно приступить к «разморозке».

Меню (рисунок 3.1) предоставляет пользователю следующие настройки «разморозки»:

– выбор времени для «заморозки», переход на момент «заморозки» и обнуление таблицы с результатами;

– выбор скорости «разморозки»;

– последовательная и хронологическая «разморозки»;

– пошаговая и автоматическая «разморозка»;

– дополнительные настройки: показ времени посылок, показ посылок совершенных в «заморозку», обработка только последней посылки.

В любой момент времени пользователь может изменять настройки «разморозки», и это никак не повлияет на работоспособность модуля подведения итогов.

Основной функцией, отвечающей за «разморозку», является getNextFunction(), которая, в зависимости от последнего действия и от настроек «разморозки», определяет, что делать дальше (приложение А, листинг А.12).

После выбора времени «заморозки», которое по умолчанию равно 4-м часам от начала соревнования, пользователь может сразу перейти к этому моменту или же запустить «разморозку» с «пустой» таблицы.

Запуск и остановка «разморозки» осуществляет нажатием клавиши «вправо» на клавиатуре. В случае, если у пользователя-неудачника нет клавиатуры, то он может запустить «разморозку» нажатием кнопки на web-странице.

В ходе «разморозки» строки в таблице сортируются по убыванию количества решенных задач. Если количество решенных задач одинаково, то строки сортируются по возрастанию штрафного времени.

Штрафное время рассчитывается как сумма по сданным задачам по следующим величинам: время от начала соревнования до первой удачной посылки по задаче, количество неудачных посылок до первой удачной умноженное на двадцать.

В ходе хронологической «разморозки» посылки обрабатываются в том порядке, в котором они были отправлены. Такую «разморозку» можно использовать для визуализации хода соревнования.

В ходе последовательной «разморозки» команды рассматриваются с последних строк таблицы к первым. В случае, если у команды имеются еще посылки – они обрабатываются, и если посылки успешные, то команда поднимается вверх по таблице. Если же посылки у команды закончились, то осуществляется переход к следующей команде. Такая «разморозка» может быть использована для подведения итогов каких-либо соревнований.

В ходе автоматической «разморозки» все действия происходят друг за другом без ожиданий каких-либо действий от пользователя. В случае если пользователь нажмёт на кнопку управления «разморозкой» – она остановится до последующего нажатия на эту же кнопку.

В ходе пошаговой «разморозки» после каждого действия модуль подведения итогов ожидает от пользователя нажатия на управляющую кнопку. Такие паузы в «разморозке» могут быть использованы, например, ведущими для комментирования хода подведения итогов соревнования или для награждения участников, занявших призовые места.

Можно выделить следующие действия, после которых ожидается нажатие на управляющую кнопку:

– подсветка команды, посылки которой рассматриваются (только при последовательной «разморозке»);

– подсветка посылки, которая обрабатывается;

– отображение результата посылки;

– перемещение команды в таблице.

Скорость «разморозки» влияет на величину паузы во время автоматической «разморозки», а так же на скорость анимации перемещения команды в таблице.

Отключение показа времени посылок может быть использовано для приведения таблицы к более компактному и информативному виду.

Дополнительная опция показа посылок заключается в том, будут ли задачи, по которым еще имеются посылки, выделяться в таблице для того, чтобы подготовить пользователя к их обработке.

Включение опции обработки только последний посылки ускорит процесс «разморозки», так как если по задаче было совершено очень много неудачных посылок, то не придётся ждать пока они все обработаются.

Выводы по разделу

Разработанная web-страница с модулем подведения итогов (рисунок 3.9) для своей стабильной работы требует:

– браузеры Google Chrome, Mozila Firefox, Opera или Internet Explorer последних версий;

– 100 мегабайт ОЗУ;

– дополнительное ПО для работы web-страницы не требуется.

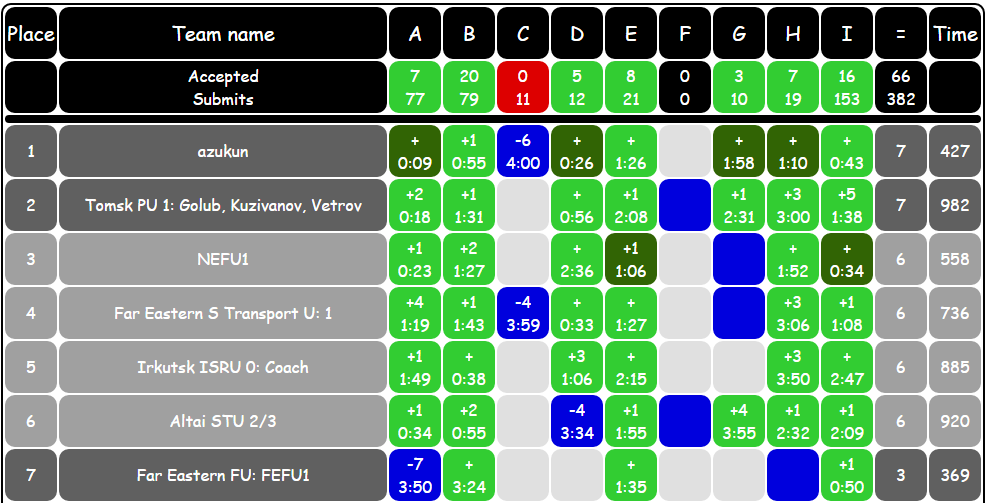


Рисунок 3.9 – Таблица с результатами соревнования по программированию

Так как модуль подведения итогов выполнен в виде web-страницы, доступ к нему осуществляется через браузер. Модуль поддерживает работу пользователей, использующих мобильные устройства.

Таким образом, посредством HTML, CSS, JavaScript, PHP, MySQL разработан модуль подведения итогов соревнований по программированию. Помимо этого использованы библиотека jQuery и технология AJAX.

3.3 Технико-экономическое обоснование

Существуют и другие модули подведения итогов, но они не имеют свободного распространения. Более того, так как их, по большей части, используют только организаторы соревнований, они не имеют удобного интерфейса. Зачастую, настройка таких модулей происходит через конфигурационный файл. Кроме того, данные модули способны обрабатывать только истории посылок определённых форматов, отличающихся от привычных для систем проведения соревнований по программированию.

Разработанный модуль подведения итогов лишен подобных минусов, и обладает рядом плюсов:

– обработка историй посылок разных форматов;

– меню с настройками «разморозки», которые можно изменять в любой момент без вреда для работоспособности модуля подведения итогов;

– возможность подключения к модулю подведения итогов с разных платформ, в том числе и мобильных.

При разработке системы для проведения соревнований по программированию с модулем подведения итогов использованы только те средства, которые находятся в способном доступе.

Заключение

В настоящее время командные чемпионаты мира по программированию являются очень востребованными, так как помогают крупным IT организациям находить новых сотрудников. Для проведения подобных чемпионатов используются специальные системы.

В ходе выполнения выпускной квалификационной работы произведён анализ существующих систем для проведения соревнований по программированию и средств для их разработки. На основе анализа спроектирована подобная система, которая обеспечивает решение следующих задач:

– проведение командных соревнований по программированию;

– автоматизация процесса тестирования студентов для сдачи практических зачетов по программированию;

– индивидуальные тренировки по программированию.

Система так же включает в себя программное обеспечение для подведения итогов соревнований по программированию, которое может использоваться вне рамок самой системы.

Система для проведения соревнований по программированию реализована на основе системы Ejudge, модуль подведения итогов разработан в виде web-страницы с использованием таких средств web-разработки, как HTML, CSS, JavaScript, PHP и MySQL, а так же с применением библиотеки jQuery и технологии AJAX.

В текущий момент, модуль подведения итогов полностью функционирует и размещен на хостинге для открытого доступа через Интернет. В то время как система для проведения соревнований по программированию установлена и настроена на тестовой машине, но в дальнейшем планируется её внедрение и использование в процессе подготовки студентов, проходящих обучение на кафедре «Вычислительная техника и компьютерная графика» «Дальневосточного государственного университета путей сообщения».

Список использованных источников

1 Contest List: Main [Электронный ресурс] – Режим доступа: http://clist.by (Дата обращения: 05.05.2016)

2 The ACM-ICPC International Collegiate Programing Contest: [Электронный ресурс] – Режим доступа: https://icpc.baylor.edu/welcome.icpc (Дата обращения: 06.05.2016)

3 VK Cup: [Электронный ресурс] – Режим доступа: http://vk.com/vkcup (Дата обращения: 06.05.2016)

4 Facebook Hacker Cup: [Электронный ресурс] – Режим доступа: https://www.facebook.com/hackercup (Дата обращения: 06.05.2016)

5 Google Code Jam: [Электронный ресурс] – Режим доступа: https://code.google.com/codejam (Дата обращения: 06.05.2016)

6 Russian Code Cup: [Электронный ресурс] – Режим доступа: http://www.russiancodecup.ru/ru/ (Дата обращения: 06.05.2016)

7 Яндекс: Алгоритм 2016: [Электронный ресурс] – Режим доступа: https://contest.yandex.ru/algorithm2016/ (Дата обращения: 06.05.2016)

8 ACM ICPC, NEERC, Southern Subregion: Порядок и правила проведения соревнований: [Электронный ресурс] – Режим доступа: https://contest.sgu.ru/?id=6 (Дата обращения: 07.05.2016)

9 SnarkNews on ACM ICPC Finals: [Электронный ресурс] – Режим доступа: http://finals.snarknews.info/ (Дата обращения: 10.05.2016)

10 Codeforces: [Электронный ресурс] – Режим доступа: http://codeforces.com/ (Дата обращения: 08.05.2016)

11 Polygon: [Электронный ресурс] – Режим доступа: https://polygon.codeforces.com/ (Дата обращения: 08.05.2016)

12 SPb IFMO Training Center: [Электронный ресурс] – Режим доступа: http://neerc.ifmo.ru/trains/information/index.html (Дата обращения: 08.05.2016)

13 Timus Onlune Judge: [Электронный ресурс] – Режим доступа: http://acm.timus.ru/?locale=ru (Дата обращения: 08.05.2016)

14 Ejudge home page: [Электронный ресурс] – Режим доступа: https://ejudge.ru/ (Дата обращения: 08.05.2016)

15 Contester: [Электронный ресурс] – Режим доступа: http://www.contester.ru/ (Дата обращения: 08.05.2016)

16 Ejudge: Общая архитектура системы: [Электронный ресурс] – Режим доступа: https://ejudge.ru/wiki/index.php/Общая\_архитектура\_системы (Дата обращения: 11.05.2016)

17 Энциклопедия сайтостроения: Статья «Технология CGI»: [Электронный ресурс] – Режим доступа: http://site.nic.ru/content/view/51/29 (Дата обращения: 11.05.2016)

18 Статья «Основы web-дизайна. С чего начать разработку сайта»: [Электронный ресурс] – Режим доступа: http://www.i2r.ru/static/255/out\_22129.shtml (Дата обращения: 13.05.2016)

19 Фримен, Э. Изучаем HTML, XHTML и CSS, 1-е издание. / Эрик Фримен, Элизабет Фримен – М.: «Питер», 2010. – 656 с.

20 Лабберс, П. HTML5 для профессионалов: мощные инструменты для разработки современных веб-приложений. / Питер Лабберс, Брайан Олберс, Фрэнк Салим – М.: «Вильямс», 2011. – 272 с.

21 Шафер, С. HTML, XHTML и CSS. Библия пользователя, 5-е издание. / Стивен Шафер – М.: «Диалектика», 2010. – 656 с.

22 Справочник CSS: [Электронный ресурс] – Режим доступа: http://htmlbook.ru/css (Дата обращения 15.05.2016)

23 Weisfeld, М. The Object-Oriented Thought Process. Fourth Edition. / Matt Weisfeld – Addison-Wesley Professional, 2013. – 336 с.

24 Страуструп, Б. Программирование: принципы и практика использования, исправленное издание. / Бьярне Страуструп – М.: Вильямс, 2011. – 1248 с.

25 Самков, Г. jQuery. Сборник рецептов. / Г. Самков – СПб.: БХВ-Петербург, 2010. – 416 с.

26 Фримен, А. jQuery для профессионалов. / Адам Фримен – М.: «Вильямс», 2012. – 960 с.

27 Хольцнер, С. Ajax Библия программиста. / Стивен Хольцнер – М.: Диалектика, 2009. – 553 с.

28 JavaScript: Введение в Ajax: [Электронный ресурс] – Режим доступа: http://javascript.ru/ajax/intro (Дата обращения: 20.05.2016)

29 Кузнецов, М. PHP на примерах. – 2-е изд. перераб. и доп. / Максим Кузнецов, Игонь Симдянов – Спб.: «БХВ-Петербург», 2011. – 400 с.

30 Зандстра, М. PHP: объекты, шаблоны и методики программирования, 3-е издание. / Мэтт Зандстра – М.: «Вильямс», 2010. – 560 с.

31 Васвани, В. MySQL: использование и администрирование. / В. Васвани – М.:[«Питер»](https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D1%82%D0%B5%D1%80_(%D0%B8%D0%B7%D0%B4%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE)), 2011. – 368 с.

32 Грофф, Д.Р. SQL: полное руководство, 3-е издание. / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель – М.: «Вильямс», 2014. – 960 с.

33 Учебная лаборатория НГУ: Ejudge testing system: [Электронный ресурс] – Режим доступа: http://swsoft.nsu.ru/~schernenok/ ejudge\_testing\_system.htm.utf8 (Дата обращения: 25.05.2016)

34 Ejudge: Инсталяция системы: [Электронный ресурс] – Режим доступа: https://ejudge.ru/wiki/index.php/Инсталляция\_системы\_ejudge (Дата обращения: 26.05.2016)

35 Гопкало, В.Н. Выпускная квалификационная работа. Общие требования и правила оформления: методическое пособие. / В.Н. Гопкало, О.А. Графский – Хабаровск: изд-во ДВГУПС, 2014. – 45 с.: ил.