Правительство Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский государственный университет»

Кафедра системного программирования

Веселков Иван Дмитриевич

Система автоматического тестирования алгоритмов с накоплением входных данных, поддержкой версионности и модулем отчетов

Дипломная работа

Допущена к защите.
Зав. кафедрой:
д. ф.-м. н., проф. Терехов А. Н.

Научный руководитель:
к. ф.-м. н. Сысоев С. С.

Рецензент:
Рубин М. С.

Санкт-Петербург
2015

SAINT-PETERSBURG STATE UNIVERSITY

Software Engineering Chair

Veselkov Ivan

Automated algorithm testing system with input data accumulation, report module and versioning support

Graduation Thesis

Admitted for defence.
Head of the chair:
Professor A. N. Terekhov

Scientific supervisor:
Sysoev Sergey

Reviewer:
Mikhail Rubin

Saint-Petersburg
2015

Оглавление

Введение

Новые технологии постоянно меняют различные аспекты нашей жизни: транспортные возможности растут, появляются новые способы коммуникации и взаимодействия между людьми. Всё это влияет на такую сферу, как здоровье: новые методы срочной эвакуации людей из катастроф, оперативное реагирование на чрезвычайные ситуации, срочное оказание первой медицинской помощи.

Помимо прочего, сейчас активно набирает обороты телемедицина - направление медицины, основанное на использовании компьютерных и телекоммуникационных технологий для обмена медицинской информацией между специалистами с целью повышения качества диагностики и лечения конкретных пациентов. Это комплексное понятие для систем, услуг и деятельности в области здравоохранения, которые могут дистанционно передаваться средствами информационных и телекоммуникационных технологий, в целях развития всемирного здравоохранения, контроля над распространением болезней, а также образования, управления и исследований в области медицины.[5] Существует множество направлений телемедицины: начиная от удаленных консультаций и заканчивая операциями на расстоянии при помощи роботов. В числе этих направлений – оперативное уведомление о критическом состоянии больных, например, диабетом, аллергиков, перенесших инфаркт или других людей, чьё здоровье и жизнь находятся в постоянной угрозе. На данный момент широко распространена так называемая "Кнопка жизни" – простое устройство в виде небольшой кнопки, которая при нажатии сразу вызывает скорую помощь.

Сейчас множество компаний разрабатывает новые устройства, которые представляют из себя простые вещи вроде расчёски, стельки для ботинка, пластырь с микросхемой и т.д. Постоянно появляются новые виды устройств.

Содержательной частью этих устройств являются датчики, которые считывают различные показатели организма, такие как температура, положение в пространстве и т.д. По данным, собираемым с датчиков, нужно получать полезную для целей диагностики информацию, такую как:

- количество шагов
- количество потребленных калорий
- количество потраченных калорий
- частота сердечных сокращений
- находится ли сейчас пациент в состоянии свободного падения
- уровень сахара в крови
- местоположение пациента
- состояние бодрствования или сна
- продолжительность и характер сна
- и многое другое

Чтобы получить подобного рода сведения, требуется нужным образом обработать показатели с датчиков специальными алгоритмами.

Эти алгоритмы постоянно развиваются и совершенствуются разработчиками, новые технологии и конфигурации оборудования для считывания показателей появляются в компаниях-разработчиках каждую неделю. Сравнение различных версий (как алгоритмов, так и оборудования) между собой требует огромное количество тестовых данных, запротоколированных в рамках экспериментов. Скорость появления новых данных растет с каждым днём.

Одной из таких компаний-разработчиков является компания Healbe.[2] В апреле 2014 они собрали почти миллион долларов на создание браслета, который собирает различную информацию, в том числе представленную выше, и который в данный момент активно развивается. По

скольку для решения каждой задачи, которых немало, требуется отдельный алгоритм, разработчики столкнулись с проблемой хранения и работы с этими данными.

Первым решением было использовать систему управления версиями: SVN или GitHub. Они позволили бы отслеживать обновления в версиях алгоритмов, хранить данные экспериментов и их протоколы, наблюдать за развитием конфигураций и вкладом тестировщиков. Однако такие системы громоздки, а их возможности избыточны. К тому же, компании нужно конфигурировать отчёты на основе тестовых данных для наглядного анализа эффективности работы алгоритмов и оборудования.

Таким образом, у компании Healbe появилась задача создать систему для хранения и работы со всеми необходимыми данными.

1. Постановка задачи

Естественным желанием разработчиков стало организовать работу со всеми данными в удобном и понятном виде: структурировано хранить версии алгоритмов, конфигурации оборудования и прочее. А так же иметь возможность наглядно сравнивать их между собой для выбора наиболее оптимальных и точных.

Таким образом было решено спроектировать систему, которая решает три основные задачи:

- Управление версиями алгоритмов, протоколами экспериментов по сбору данных и конфигурациями оборудования
- Формирование настраиваемых отчётов для выбора лучших алгоритмов и конфигураций
- Универсальность: запуск и обработка результатов разных алгоритмов должны быть одинаковыми и максимально простыми, но не в ущерб возможностям конфигуратора отчётов

2. Описание решения

2.1. Требования

2.1.1. К системе в целом

- Система должна быть централизованной, т.е. все данные хранятся в центральном хранилище
- Хранилище состоит из базы данных и файловой системы:
  - В базе данных необходимо хранить:
    * зависимости данных и алгоритмов, которые использовались для их сбора
    * техническая информация (дата и время загрузки на сервер, пользователь загрузивший файлы, путь к ним и т.д.)
    * другая информация необходимая для работы с файлами (например, запускающие инструкции для алгоритмов или тестировщик, который проводил эксперимент, для экспериментов)
  - Файловая система должна обеспечивать хранение:
    * готовые к запуску версии алгоритмов со всеми необходимыми зависимостями
    * текстовые файлы экспериментов
- Доступ пользователя к системе осуществляется через веб-интерфейс
- Смежными системами являются:
  - база данных компании с ФИО тестировщиков
  - база данных компании с описаниями конфигураций оборудования

2.1.2. К функциям системы

- Система авторизации пользователя
  - Разделяет пользователей на группу аналитиков и группу администраторов
  - Подробнее о возможностях групп описано в разделе (2.4)
- Система для работы с:
  - протокольными данными экспериментов
    * Загрузка новых данных
    * Редактирование и удаление данных на сервере
  - версиями алгоритмов
    * Загрузка новых версий алгоритмов
    * Редактирование и удаление уже загруженных версий
    * Запуск алгоритмов с помощью универсального интерфейса
  - отчётами
    * Конфигурация различных форм отчётов
    * Наглядный интерфейс для обработки и анализа результата работы
- Система доступа к базам
  - с тестировщиками компании
  - с конфигурациями тестируемого оборудования

2.2. Технологии

- Уровень хранения данных
  - Windows
  - MySQL[3]
- Уровень сервера приложения
  - Django[4]
- Уровень представления пользователю
  - Twitter Bootsrap[1]
  - jQuery[6]

2.3. Модель

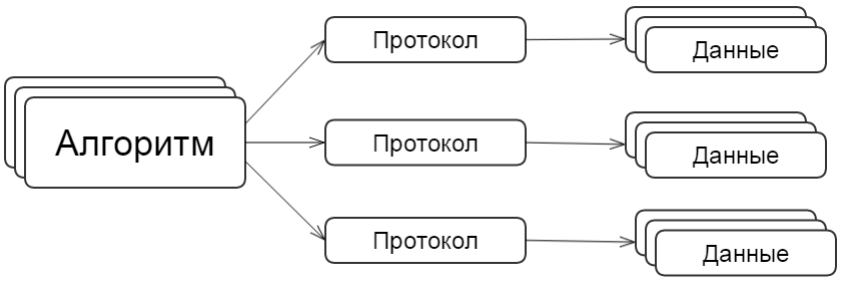


Рис. 1: Наглядное представление взаимодействия некоторого алгоритма и тестовых данных

Каждый алгоритм предназначен для решения определенной задачи или цели. И для каждого из них существуют различные протоколы экспериментов, содержащие некий набор данных тестирования.

Для описания этой модели в базе данных были реализованы классы, на основе которой фреймворк автоматически построил структуру базы данных:

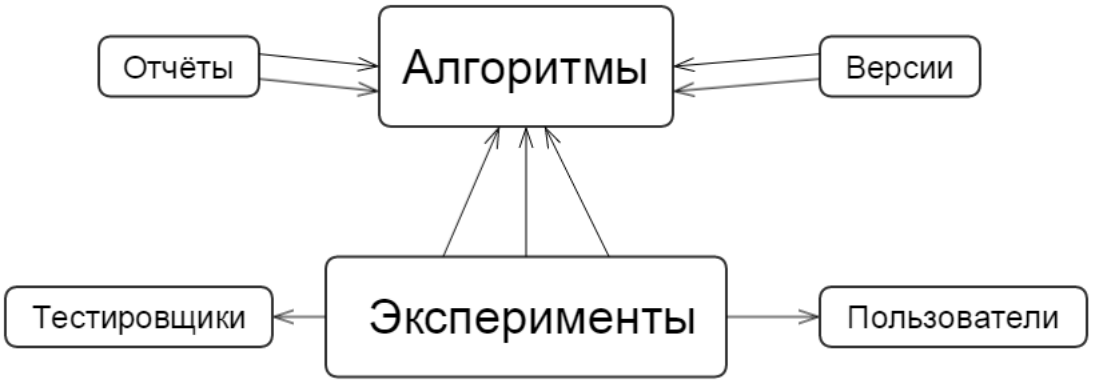


Рис. 2: Структура таблиц базы данных

Таблицы:

- Алгоритмы, с указанием задачи, которую он решает
- Версии, каждая из которых содержит путь к папке с программой и связана со своим алгоритмом
- Отчёты, каждый из которых описывает форматы отчётов для некоторого алгоритма (см. рис. 2.5.1)
- Пользователи системы. С разделением на группы администраторов и аналитиков (см. подраздел 2.4)
- Тестировщики компании, которые проводили эксперименты по сбору данных
- Эксперименты содержат следующие поля:
  - Путь к файлу-протоколу с тестовыми данными
  - ID тестировщика, проводившего эксперимент
  - ID пользователя, загрузившего эксперимент в систему
  - ID алгоритма, тестирование которого проводилось
  - Дата создания и дата последнего изменения записи

2.3.1. Универсальность протоколов

Каждый из алгоритмов требует от протоколов некие данные необходимые ему для работы. Для хранения этих протокольных данных в базе и для выполнения задачи универсальности (пункт 3 раздела 1) были предложены следующие решения:

1. Для каждого алгоритма заводить свою отдельную таблицу протоколов, содержащую данные в том формате, которые будут использоваться программой.
   Но это решение трудоемкое и, помимо прочего, требовало бы дополнительных изменений структуры базы данных при добавлении нового алгоритма

2. Зарезервировать несколько различных типов полей для каждого протокола эксперимента, которые будут использованы для записи необходимых протокольных данных при добавлении новых экспериментов.
   Однако тогда они все будут предлагаться для заполнения в форме создания нового эксперимент, хотя далеко не все из них будут использоваться. К тому же пользователю каждый раз вспоминать и оценивать алгоритм, чтобы понять для чего предназначено каждое из полей.

3. Гибридный вариант: для каждого алгоритма есть определенный набор зарезервированных параметров, по три штуки каждого из основных типов: Integer (целое число, точность 32 бит), Float (число с плавающей запятой, точность 64 бит) и String (строковый параметр, максимальная длина – 255 символов). При создании в системе нового алгоритма, предлагается указать названия для параметров, которые необходимы для протоколов, связанных с этим типом экспериментов. Эти названия будут предложены для заполнения при загрузке новых данных эксперимента, связанных с этим алгоритмом.

По итогам рассмотрения приведенных выше решений, был реализован именно гибридный вариант. Ниже представлена реализация решения в приложении:

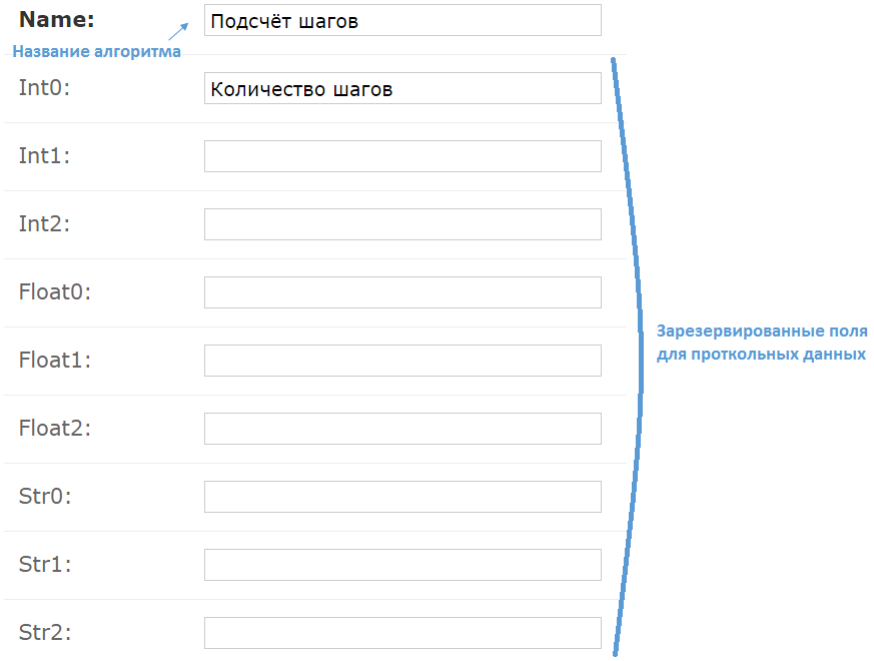


Рис. 3: Пример формы описания алгоритма

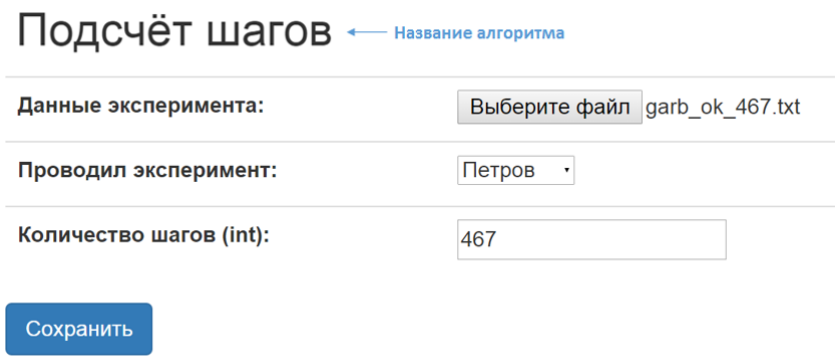


Рис. 4: Пример формы загрузки данных эксперимента

2.4. Система

Работать в системе будут два типа пользователей: аналитики и администраторы.

Возможности администраторов:

- Добавлять новых пользователей и администраторов
- Загружать версии алгоритмов и конфигурации оборудования
- Создавать новые шаблоны отчётов (см. рис. 2.5.1)
- Изменять и удалять уже загруженные элементы

Также они имеют доступ к возможностям аналитиков, речь о которых пойдёт ниже.

Возможности аналитиков:

- Загружать новые данные запротоколированных экспериментов
- Изменить и удалять уже загруженные данные
- Запускать сравнение различных версий алгоритмов и оборудования

В первую очередь выбирается цель, которую решают алгоритмы, которые мы хотим сравнивать. Затем предлагается выбрать протоколы экспериментов и версии этого алгоритма. Также выбирается формат отчётов.

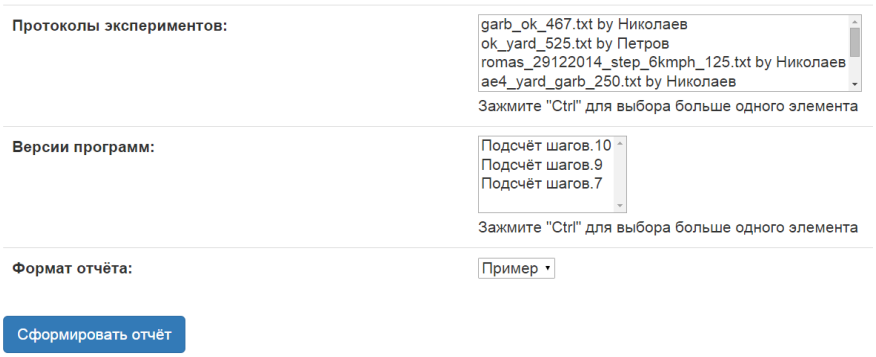
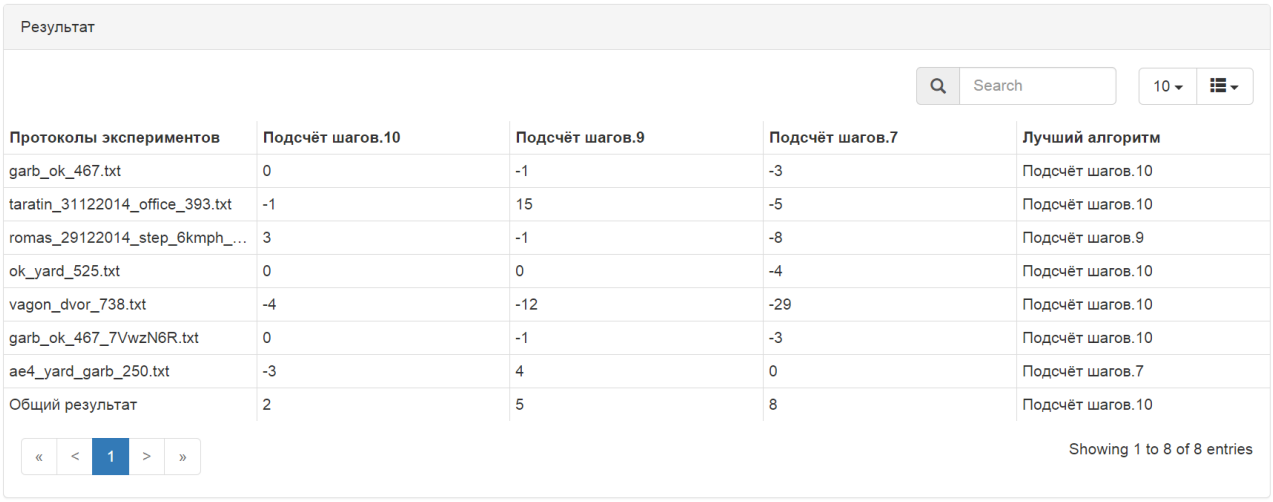


Рис. 5: Пример страницы формирования отчёта

2.5. Отчёт

Было реализовано представление отчёта в формате таблицы, строками в которой являются протоколы экспериментов, столбцами – версии алгоритмов. В ячейках расположен результат работы программы данной версии над конкретными данными эксперимента, а в конце каждого столбца приведён некий общий (суммирующий) результат по каждой из версий. Формат этих результатов (для каждого эксперимента и общий) описывался в шаблоне отчёта. В самом крайнем столбце выписан лучший алгоритм по каждому из экспериментов и по общим результатам.


Результат

Search 10

Протоколы экспериментов	Подсчёт шагов.10	Подсчёт шагов.9	Подсчёт шагов.7	Лучший алгоритм
garb_ok_467.txt	0	-1	-3	Подсчёт шагов.10
taratin_31122014_office_393.txt	-1	15	-5	Подсчёт шагов.10
romas_29122014_step_6kmph_...	3	-1	-8	Подсчёт шагов.9
ok_yard_525.txt	0	0	-4	Подсчёт шагов.10
vagon_dvor_738.txt	-4	-12	-29	Подсчёт шагов.10
garb_ok_467_7VwzN6R.txt	0	-1	-3	Подсчёт шагов.10
ae4_yard_garb_250.txt	-3	4	0	Подсчёт шагов.7
Общий результат	2	5	8	Подсчёт шагов.10

« < 1 > »

Showing 1 to 8 of 8 entries

Рис. 6: Пример отчёта. В ячейках – процент отклонения результатов вычисления от предполагаемых. В общем результате по каждой из версии – средний процент отклонения.

2.5.1. Требования к формированию отчётов

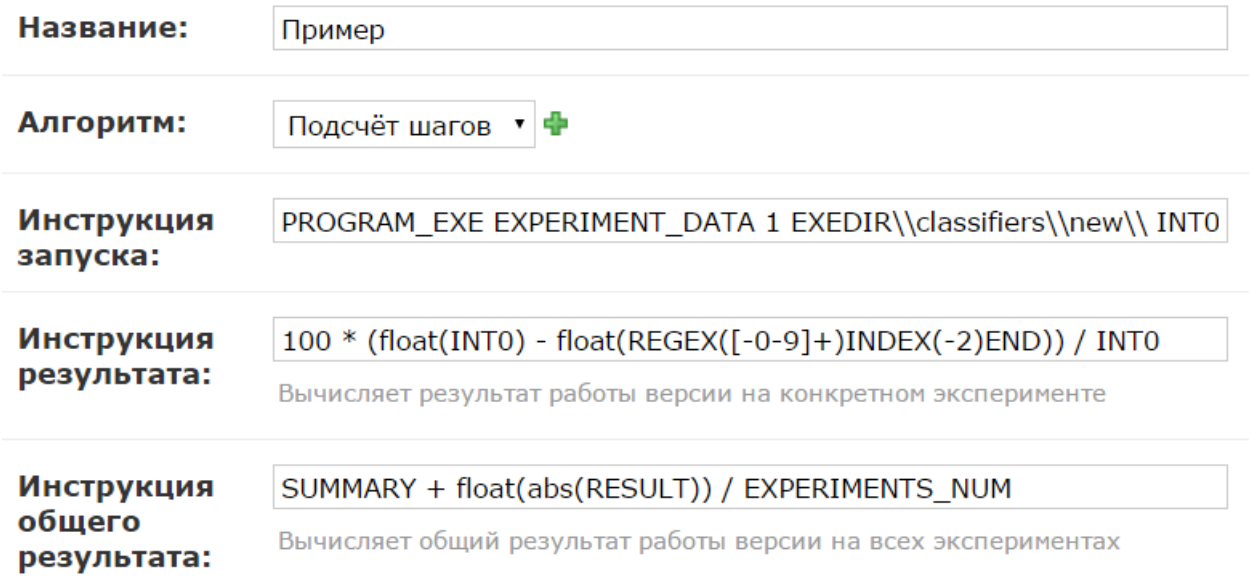


Рис. 7: Пример формы отчёта

Для создания отчёта требуется определить:

- название формы для её идентификации
- алгоритм, на который настроен этот отчёт
- инструкцию запуска программы
- инструкцию обработки результата, которая вычисляет результат работы версии алгоритма на конкретном протоколе эксперимента
- инструкцию обработки общего результата, которая вычисляет общий результат работы версии алгоритма на всех экспериментах

Рассмотрим эти инструкции подробнее:

2.5.2. Инструкция запуска

При описании шаблона отчёта, требуется заполнить инструкцию, которая будет выполняться для запуска программы. В ней, с помощью ключевых слов, указываются необходимые параметры, которые при формировании отчёта будут заменены на соответсвующие пути, взятые из базы данных.

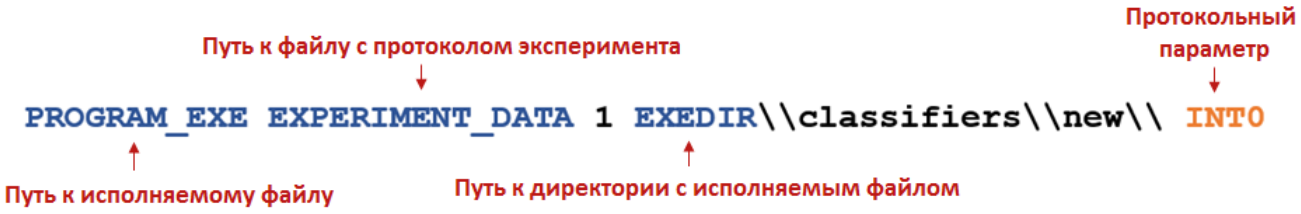


Рис. 8: Пример запускающей инструкции

Ключевые слова:

- PROGRAM_EXE – путь к программе, версии алгоритма
- EXPERIMENT_DATA – путь к текстовому файлу протокола эксперимента
- EXEDIR – путь к директории с программой. Используется для указания в качестве входного параметра различных зависимостей, необходимых для работы
- INT0, INT1, INT2, FLOAT0, ..., STRING2 – зарезервированные в записи алгоритма поля для протокольных данных. Берутся из записи протокола в базе данных

2.5.3. Инструкция результата

Так как сравнивать алгоритмы можно по разным результирующим параметрам, в отчёте нужно указать инструкцию для формирования результата. Инструкция может содержать:

- Числа: 1337, 987.12
- Строки: ”example”
- Простейшие арифметические операции $+ - */$
- Целочисленное возведение в степень: 2**3 = 8
- Приведение типов: int(2.0) => 2, float(2) = 2.0, string(2.0) = ”2.0”
- Функция модуля: abs(-2) = 2
- Конкатенация строк: ”ab” + ”c” = ”abc”
- Регулярные выражения

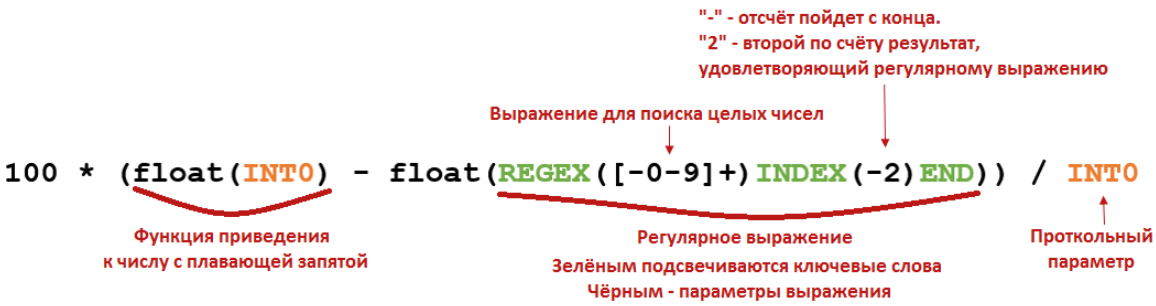


Рис. 9: Пример инструкции вычисления результата работы. Здесь используется один из входных параметров и регулярное выражение, для получения данных из вывода программы

Заключение

В результате проделанной работы:

- Спроектирована и реализована, с учётом всех необходимых требований, универсальная система сравнения версий программных продуктов
- Система протестирована на различных алгоритмах
- В данный момент система проходит внедрение в компании Healbe

Список литературы

[1] Bootstrap community. Bootstrap // Официальный сайт. — 2012. — http://goo.gl/03ur8.

[2] Healbe Corp. Блог компании // Официальный сайт. — 2014. — https://goo.gl/1D5hO4.

[3] Wikipedia. MySQL // Википедия, свободная энциклопедия. — 2004. — https://goo.gl/6j1P2G.

[4] Wikipedia. Django // Википедия, свободная энциклопедия. — 2007. — https://goo.gl/iBvqqB.

[5] Wikipedia. Телемедицина // Википедия, свободная энциклопедия. — 2013. — https://goo.gl/0zGAXK.

[6] jQuery foundation. jQuery // Официальный сайт. — 2012. — https://goo.gl/c4hetS.