МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ (ФИЛИАЛ)
ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»
В Г. ТАГАНРОГЕ РОСТОВСКОЙ ОБЛАСТИ
ПИ (филиал) ДГТУ в г. Таганроге
ЦМК «Прикладная информатика»

Практикум

По выполнению практических работ
по профессиональному модулю:
ПМ. 03. «Ревьюирование программных продуктов»

основной профессиональной образовательной программы (ОПОП)

по специальности 09.02.07

«Информационные системы и программирование»

Таганрог

2023 г.

Составители: Андриян О.В.

Практикум по выполнению практической работы по профессиональному модулю: ПМ. 03. «Ревьюирование программных продуктов».

ПИ (филиала) ДГТУ в г.Таганроге, 2023г.

В практикуме кратко изложены теоретические вопросы, необходимые для успешного выполнения практической работы, рабочее задание и контрольные вопросы для самопроверки.

Предназначено для обучающихся по специальности 09.02.07 «Информационные системы и программирование».

Ответственный за выпуск:

Председатель ЦМК: _____ О.В. Андриян

Содержание

1	Область применения методических указаний	5
2	Основные задачи проведения практических работ	5
3	Организация и проведение практических работ	8
4	Практические работы	8
5	Индивидуальное задание	8
6	Основные показатели оценки результата и их критерии	8
7	Перечень использованных информационных ресурсов	10
8	Приложения	10

1 Область применения методических указаний (рекомендаций)

Методические рекомендации предназначены в качестве методического пособия при проведении практических работ по учебному предмету (модулю), практике и государственной итоговой аттестации для специальности СПО.

Практические работы проводятся после изучения соответствующих разделов и тем по учебному предмету (модулю), практике и государственной итоговой аттестации. Выполнение обучающимися практических работ позволяет им понять, где и когда изучаемые теоретические положения, и практические умения могут быть использованы в будущей практической деятельности.

2 Основные задачи проведения практических работ

Цель:

− формирование практических умений, необходимых в последующей профессиональной и учебной деятельности.

Задачи:

− обобщить, систематизировать, углубить, закрепить полученные теоретические знания по конкретным темам дисциплин общепрофессионального цикла;

− формировать умения применять полученные знания на практике;

− выработать при решении поставленных задач таких профессионально значимых качеств, как самостоятельность, ответственность, точность, творческая инициатива.

На практических занятиях обучающиеся овладевают первоначальными профессиональными умениями и навыками, которые в дальнейшем закрепляются и совершенствуются в процессе учебной и производственной практики.

Освоение дисциплины является частью освоения основного вида профессиональной деятельности и соответствующих общих (ОК) компетенций:

Рабочая программа учебной дисциплины является частью основной образовательной программы в соответствии с ФГОС по специальности СПО, 09.02.07 Информационные системы и программирование в части формирования соответствующих компетенций:

ОК 01. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.

ОК 02. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.

ОК 04. Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.

ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста

ПК 11.1. Осуществлять сбор, обработку и анализ информации для проектирования баз данных

ПК 11.2. Проектировать базу данных на основе анализа предметной области.

ПК 11.3. Разрабатывать объекты базы данных в соответствии с результатами анализа предметной области.

ПК 11.4. Реализовывать базу данных в конкретной системе управления базами данных.

ПК 11.5. Администрировать базы данных

В результате выполнения практических работ, предусмотренных программой по учебному предмету (модулю), практике и государственной итоговой аттестации, обучающийся должен:

Уметь:

- строить информационную модель данных для конкретной задачи;
- выполнять нормализацию базы данных;
- подбирать наилучшую систему управления базами данных (СУБД)
- проектировать прикладную информатику

Знать:

- строить информационную модель данных;
- типы логических моделей;
- этапы проектирования базы данных
- общую теорию проектирования прикладной программы. Коды формирующих компетенций:

О проведении практической работы обучающимся сообщается заблаговременно: когда предстоит практическая работа, какие вопросы нужно повторить, чтобы ее выполнить. Просматриваются задания, оговаривается ее объем и время ее выполнения. Критерии оценки сообщаются перед выполнением каждой практической работы.

Перед выполнением практической работы повторяются правила техники безопасности. При выполнении практической (лабораторной) работы обучающийся придерживается следующего алгоритма:

1. Записать дату, тему и цель работы.

2. Ознакомиться с правилами и условиями выполнения практического (лабораторного) задания.

3. Повторить теоретические задания, необходимые для рациональной работы и других практических действий.

4. Выполнить работу по предложенному алгоритму действий.

5. Обобщить результаты работы, сформулировать выводы по работе.

6. Дать ответы на контрольные вопросы.

7. Объем может колебаться в пределах **5-10 печатных страниц, в зависимости от работы** (***оформление письменной работы согласно Правилам оформления письменных работ обучающихся для гуманитарных/технических направлений подготовки (приказ ректора Б.Ч. Месхи от 16.12.2020 года №242***); все приложения к работе не входят в ее объем.

Работа должна быть выполнена грамотно, с соблюдением культуры изложения.

Обязательно должны иметься ссылки на использованные информационные источники. Должна быть соблюдена последовательность написания библиографического аппарата.

Задание со звездочкой повышенной сложности на оценку **«отлично».**

Практическое занятие №1 «Создание и изучение возможностей репозитория проекта»

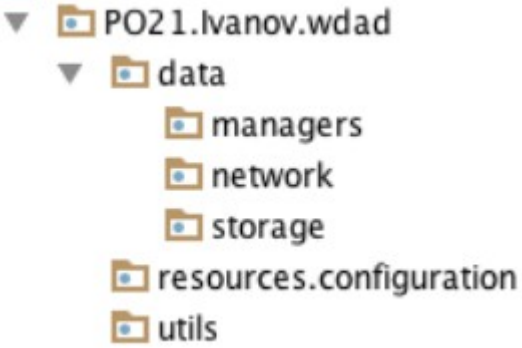
Цель работы: научиться создавать репозиторий проекта, ознакомиться с возможностями

Форма отчета:
−выполнить задание;
−показать преподавателю;
−ответить на вопросы преподавателя.
Время выполнения: 2 ч

Задание 1 Зарегистрируйтесь в сервисе GitHub.

Задание 2 Установите VCS git на свой компьютер и настройте IDE для работы с данной системой. Создайте новый проект «starting-monkey-to-human-path».

Задание 3 Создайте пакет «..wdad». Внутри него создайте следующую иерархию пакетов



Задание 4 Создайте новый репозиторий git и установите проект под версионный контроль. Создайте новый удаленный репозиторий на GitHub из локального (init и commit&push)). Добавьте в соавторы (collaborator) пользователя WildChildCode.

Задание 5 Создайте новую ветку (branch) «initTask». Добавьте следующие изменения в рамках этой ветки. В пакете wdad создайте класс Application. Этот класс должен содержать метод public static void main(String[] args), выводящий в консоль фразу: «I’m , and I’m not a monkey» Зафиксируйте изменения в локальном и удаленном репозиториях (commit & push). Коммит снабдите комментарием: Added entry point

Задание 6 На страничке удаленного репозитория на основе последнего коммита создайте пул реквест (Сравниваются ветки master и initTask). В комментарии при создании пул реквеста (вкладка write) добавьте текст в следующем формате: “ инициализацию успешно прошел (-ла). Готов (-а) приступить к выполнению следующего задания”

Практическое занятие № 2 «Экспорт настроек в командной среде разработки»

Цель работы: научиться экспортировать настройки в командной среде разработки

Форма отчета:
−выполнить задание;
−показать преподавателю;
−ответить на вопросы преподавателя.
Время выполнения: 2 ч

ТЕОРИЯ

1. Информация о командной строке Windows.
Командная среда - это программный продукт Microsoft, который обеспечивает связь между пользователем компьютера и операционной системой. Командная оболочка Windows использует интерпретатор команд cmd.exe и присутствует во всех версиях операционных систем Windows. Многие возможности и функции управления операционной системой недоступны из графического интерфейса и поэтому cmd является единственным средством доступа к этим инструментам.
Отличием работы из командной строки является полное отсутствие больших и громоздких графических утилит. Пользовательский интерфейс текстовой строки предоставляет среду, в которой выполняются приложения и служебные программы. Среда, эмулирующая DOS имеет множество названий, таких как – командная строка, окно, среда и .т.д.

1.1.Преимущества командной строки Windows
Для просто обывателя слово «командная строка» звучит без особого смысла и автоматически относится к лексикону профессионалов компьютерного мира, что, по сути, так и есть, так как командную строку используют, преимущественно профессионалы. Почему знатоки компьютерного дела чаще выбирают командную строку вместо других, объяснить легко, так как командная строка предоставляет следующие возможности:

- С помощью cmd возможно создание сценариев автоматизации и пакетных файлов, т.е. выполнение одной или нескольких команд без вмешательства пользователя. Это отличный инструмент для создания сценариев, а также вы сможете в полной мере использовать команды для управления реестром. Это значит, что одна или несколько команд будут выполняться без какого-либо вмешательства пользователя. Одним из примеров автоматической работы программного обеспечения служит настройка на автоматическое открытие необходимых вам программ при включении компьютера;
- Управление данными и файлами. Преимущества cmd становятся очевидны, когда требуется выполнять однотипные операции над множеством объектов. Одним из важных преимуществ командной строки является непосредственная возможность командной строки управлять файлами и данными. К данным возможностям относятся: копирование, удаление, перемещение и т.д. При этом, не забывайте, что вы можете автоматизировать данный процесс.
- Администрирование компьютера. Быстрое получение текущей информации сокращает время диагностики компьютера.
- Администрирование сети. Многие команды администрирования сети не имеют графических эквивалентов (например – команда ping, pathping, tracert). Командная строка очень удобна для контроля сетевой активности. Вы можете создавать службы, запускающиеся при старте оперативной системы, можете использовать команды администрирования сети, не имеющие графических эквивалентов;

По сравнению со многими другими интерфейсами между пользователем и компьютером имеет текстовую среду вместо графической, что очень полезно в управлении программами, так как экономится значительное количество времени.

2. Файлы, каталоги, файловая система - дерево каталогов. Практически вся информация на компьютерах представлена в виде файлов. Файл является основной единицей хранения данных и программ обрабатывающих эти данные. Файл - это именованная (т.е. снабженная именем) область внешней памяти. Операционная система и прикладные программы (приложения) получают доступ к файлу по его имени. Максимальная длина имени файла или каталога в Windows 256 символов, включая расширение, имя и расширение разделяются точкой. Расширение указывает на вид информации или на приложение, которым может быть открыт этот файл, например myfile.txt - текстовый файл, myfile.doc - документ MS Word и т.д.

2.1. Дерево каталогов

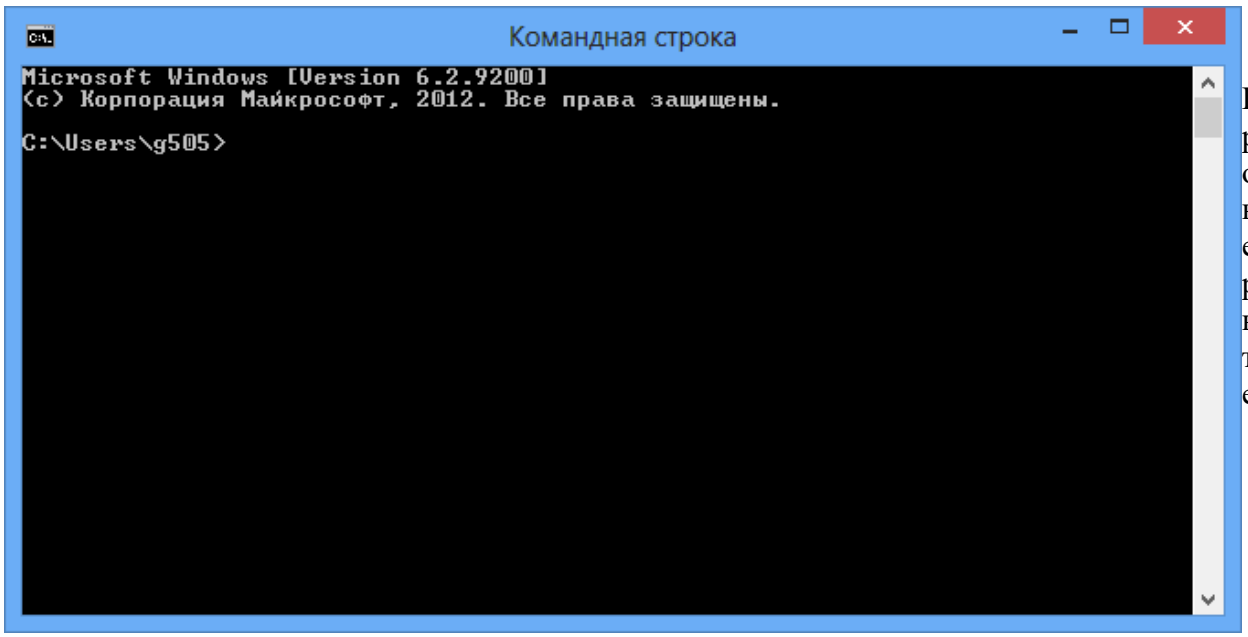
Файлы хранятся в системе вложенных каталогов (директорий) и организуются в файловую систему. Таким образом, файловой системой называется совокупность файлов и каталогов, организованных в древовидную структуру. Ее можно представить как перевернутое вверх корнем дерево. Узлами, из которых расходятся "ветви", являются каталоги, восходящие, в конечном счете, к корневому каталогу. Узлам, из которых не происходит дальнейшего ветвления, как правило, соответствуют файлы, хотя это могут быть и пустые каталоги. Обычно мы говорим: "Каталог (директория) содержит файлы" или "Файл находится в каталоге". Но при этом понимаем, что каталог не является областью памяти, собственно вмещающей сами файлы. Каталог лишь содержит список файлов, с указанием их имен и других атрибутов. По сути, каталог - это специализированный файл, назначением которого является хранение списка отнесенных к нему файлов (в том числе и подкаталогов, которые, как и все каталоги - тоже файлы). "Пустая" файловая система состоит только из корневого каталога.

2.2. Рабочий каталог. Текущий каталог, абсолютный путь и относительный путь. Когда мы входим в систему, то попадаем в свой рабочий каталог, он задан для нас системным администратором и сами мы его поменять не можем. В момент входа он является так же текущим каталогом. Текущим мы можем сделать любой существующий каталог, к которому у нас есть доступ. Зачем нужен текущий каталог? Дело в том, что операционная система осуществляет доступ к файлу или каталогу через его путевое имя или, попросту путь. Существуют два вида путей, абсолютные и относительные. Абсолютный путь - это последовательность имен каталогов, которая начинается от корневого каталога и, следуя по дереву файловой системы, заканчивается именем каталога или файла, с которым мы хотим работать. Относительный путь может быть указан от текущего или рабочего каталога, что может оказаться значительно короче и удобней, чем использование абсолютного пути. Когда мы находимся в рабочем каталоге, нам не нужно указывать пути к каталагам и файлам находящимся в нем. К другим каталогам придется указать путь, но не абсолютный, а гораздо более короткий. Но об этом немного позднее, а пока приступим к работе.

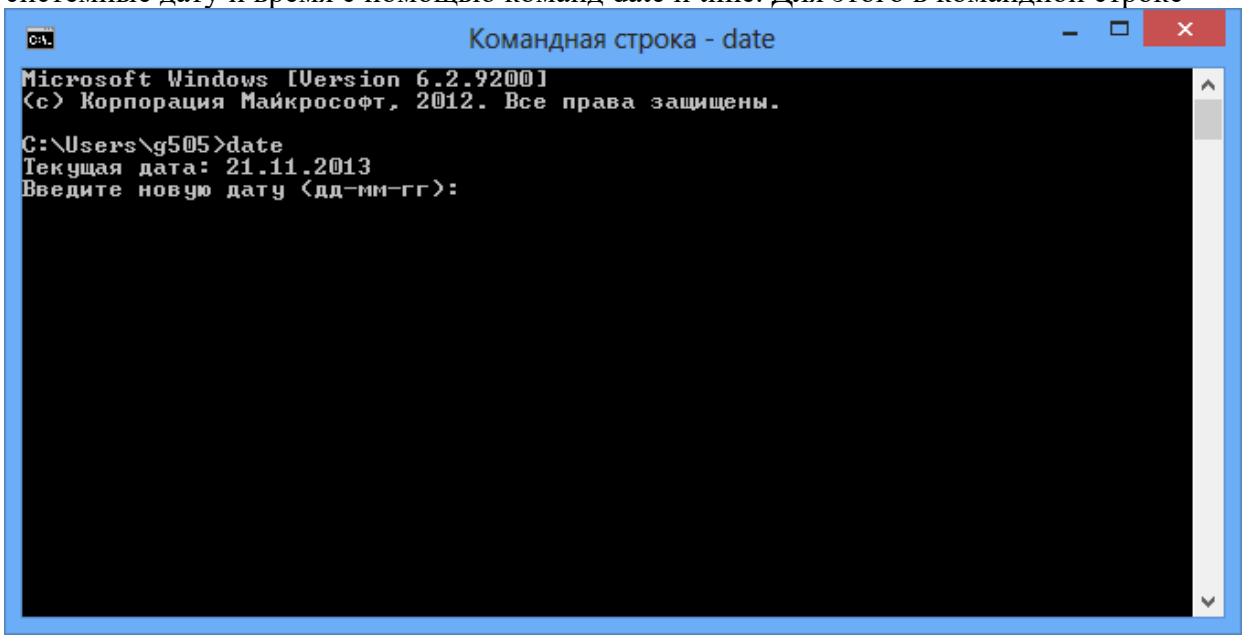
3. Полезная функция. Попробуйте понажимать клавиши "ВВЕРХ" и "ВНИЗ" на клавиатуре, и Вы увидите, что командная строка запоминает команды, и этими самыми стрелочками их можно перелистывать и, при необходимости редактировать.

ЗАДАНИЕ

1. Создайте на рабочем столе папку Test.
2. Запустите командную строку cmd.exe (Пуск — ввод с клавиатуры «cmd» без кавычек).

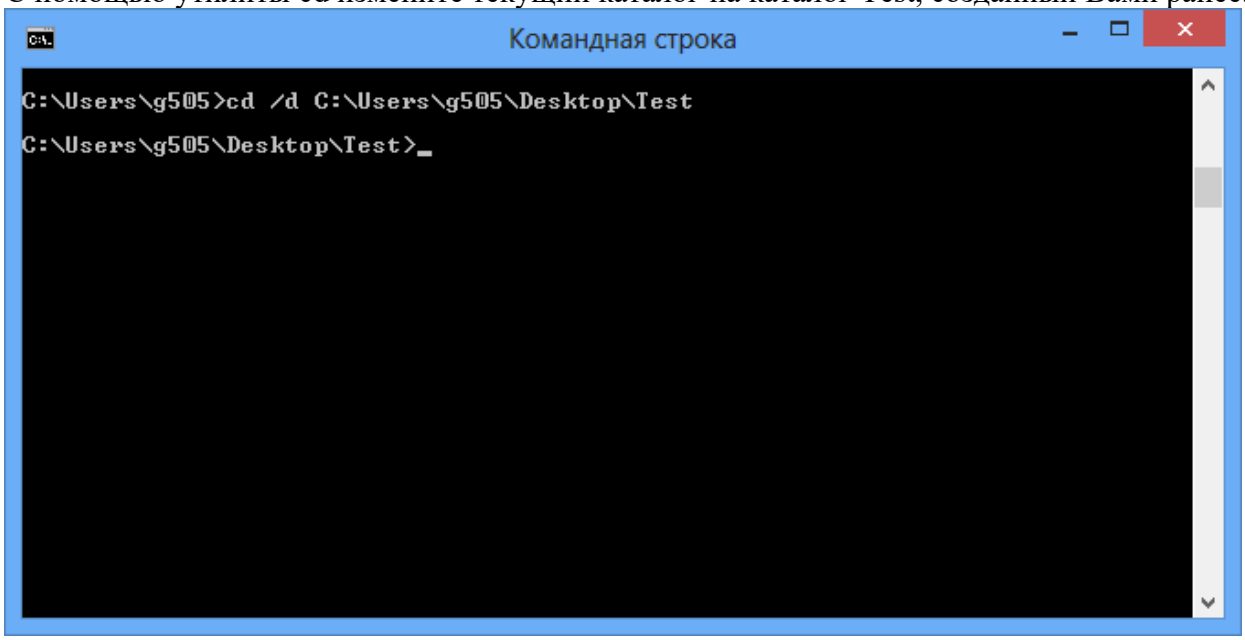
3. 

Проверьте системные дату и время с помощью команд date и time. Для этого в командной строке

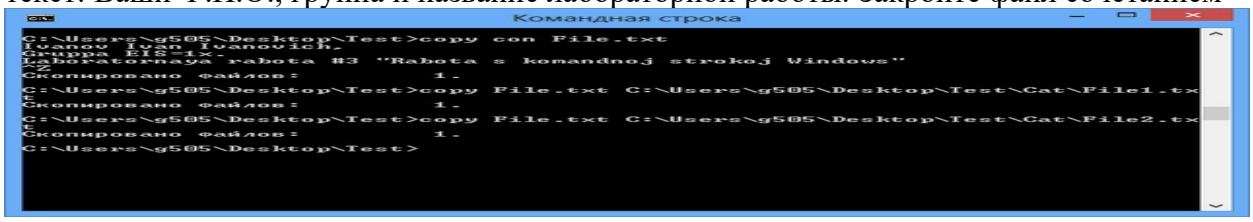


наберите нужную команду и нажмите Enter.

4. С помощью утилиты cd измените текущий каталог на каталог Test, созданный Вами ранее.

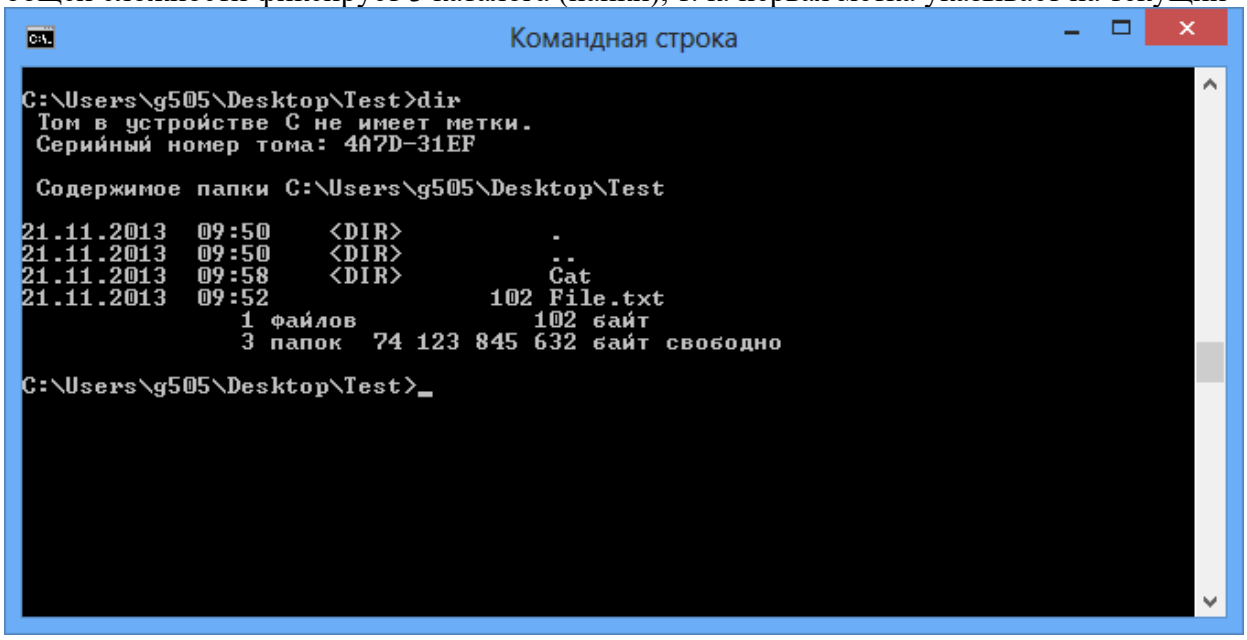


5. С помощью команды md создайте каталог с именем Cat.
6. Используя команду copy con, создайте файл с именем File.txt. (Команда copy con означает копирование с консоли, т. е. с клавиатуры). После данной команды введите следующий текст: Ваши Ф.И.О., группа и название лабораторной работы. Закройте файл сочетанием



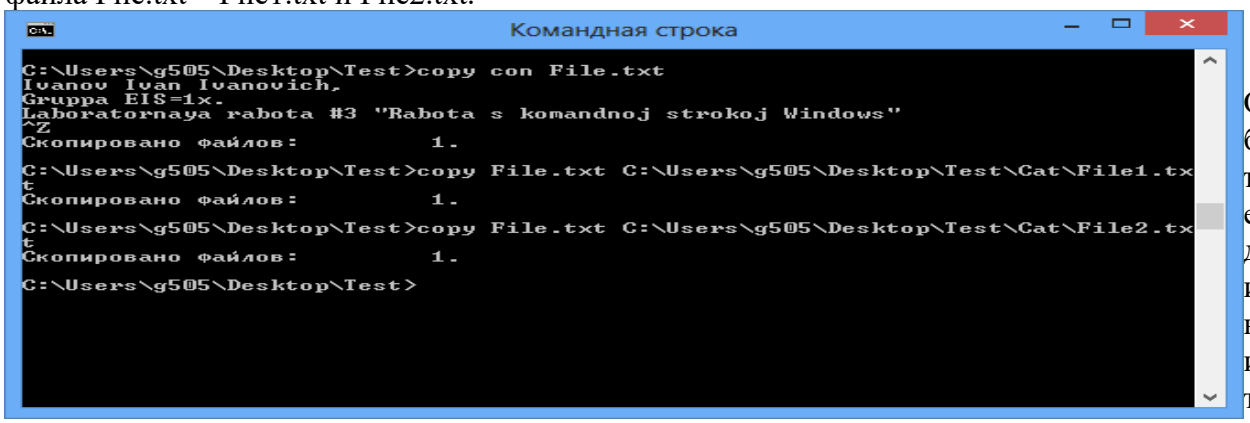
клавиш Ctrl+Z.

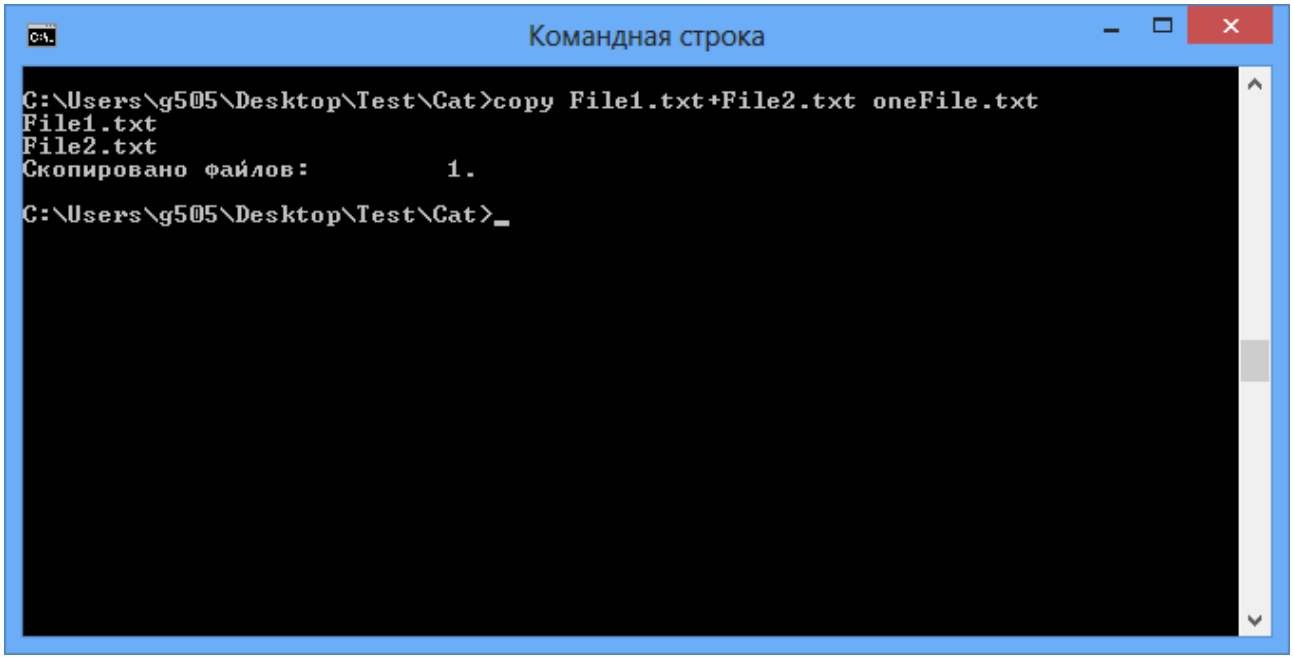
7. С помощью команды dir просмотрите список созданных объектов в папке Test. Команда в общей сложности фиксирует 3 каталога (папки), т. к. первая метка указывает на текущий



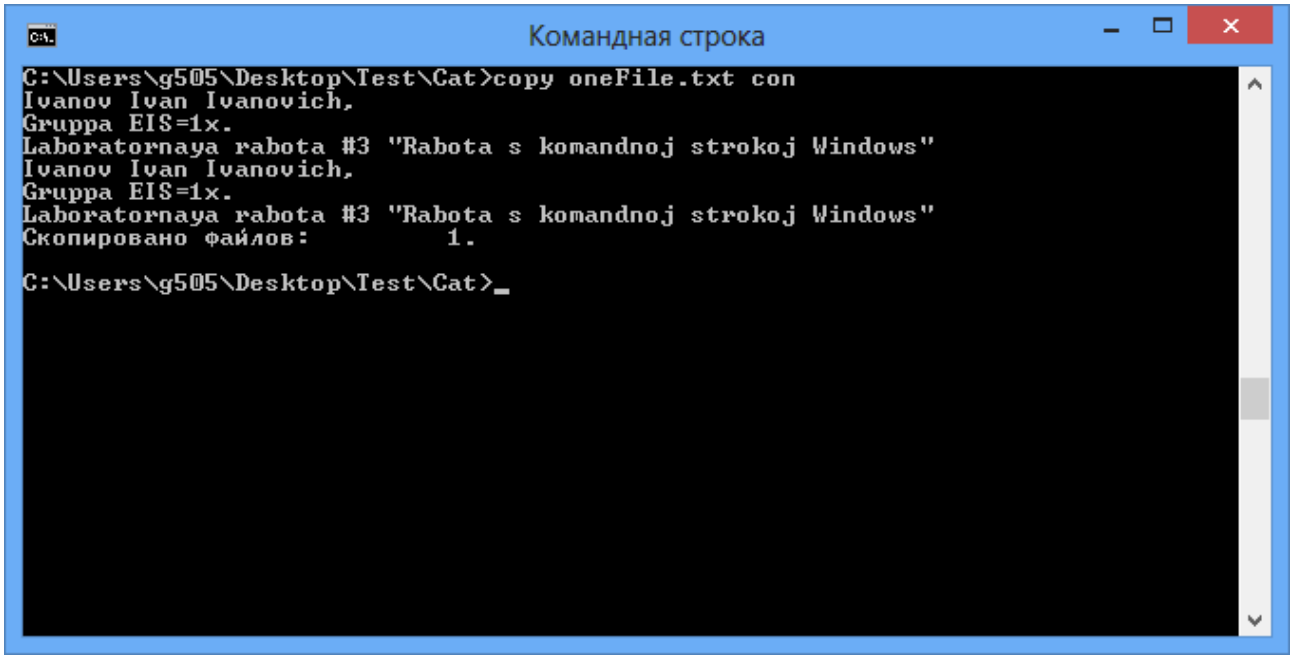
каталог, обозначенный точкой, вторая — на предыдущий каталог (две точки).

8. В каталоге Cat с помощью команды copy создайте две копии файла File.txt – File1.txt и File2.txt.
9. Объедините файлы File1.txt и File2.txt в файл oneFile.txt с помощью команды copy.



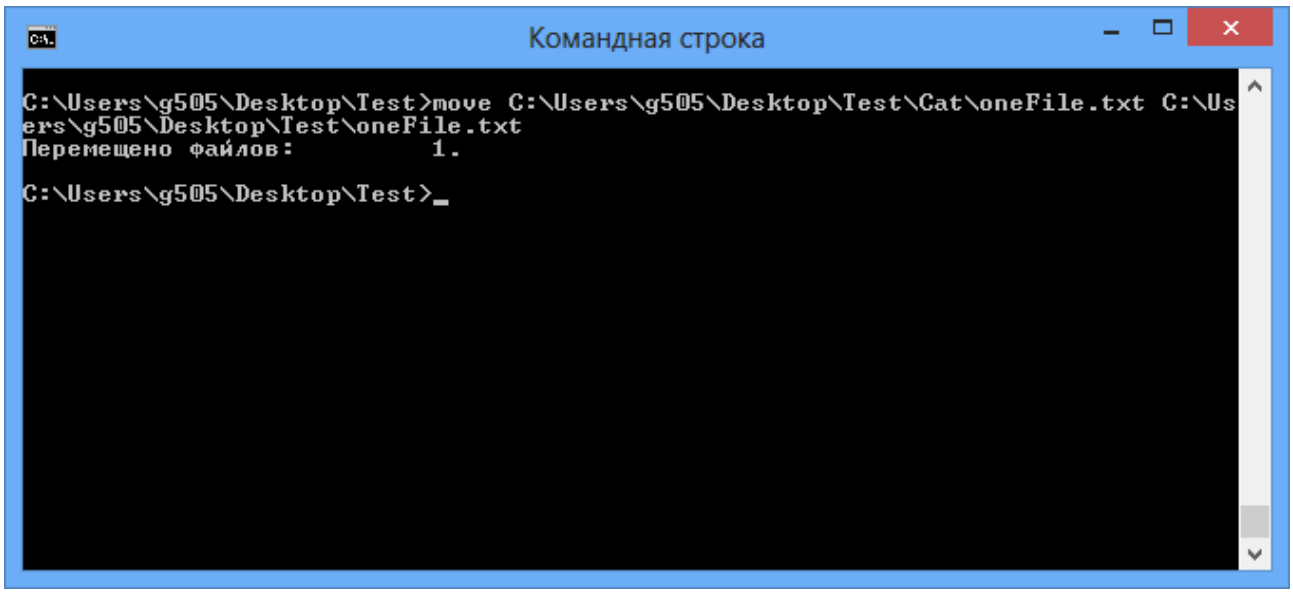


10.

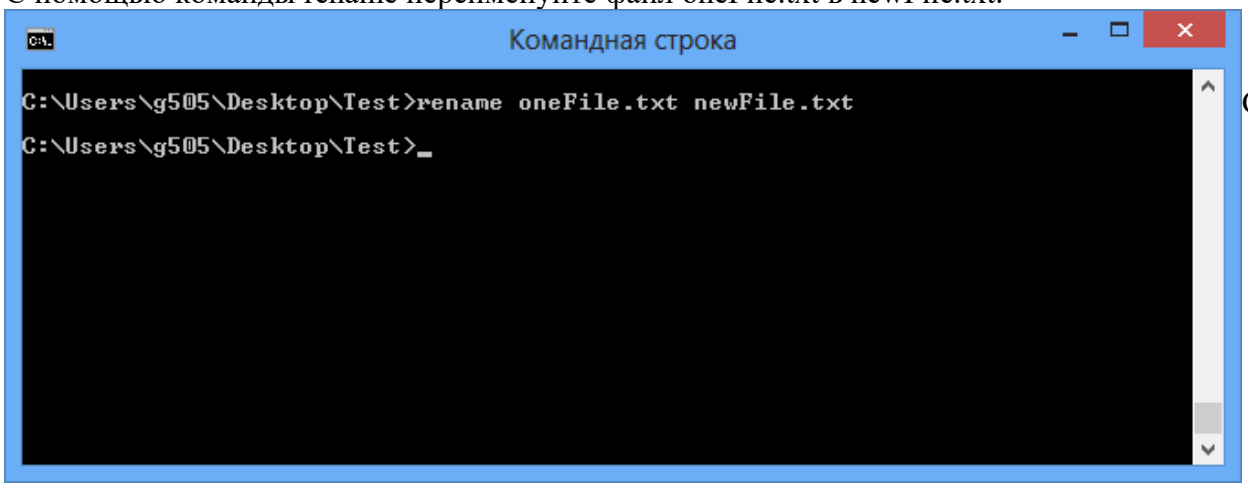


Просмотрите полученный файл oneFile с помощью утилиты copy.

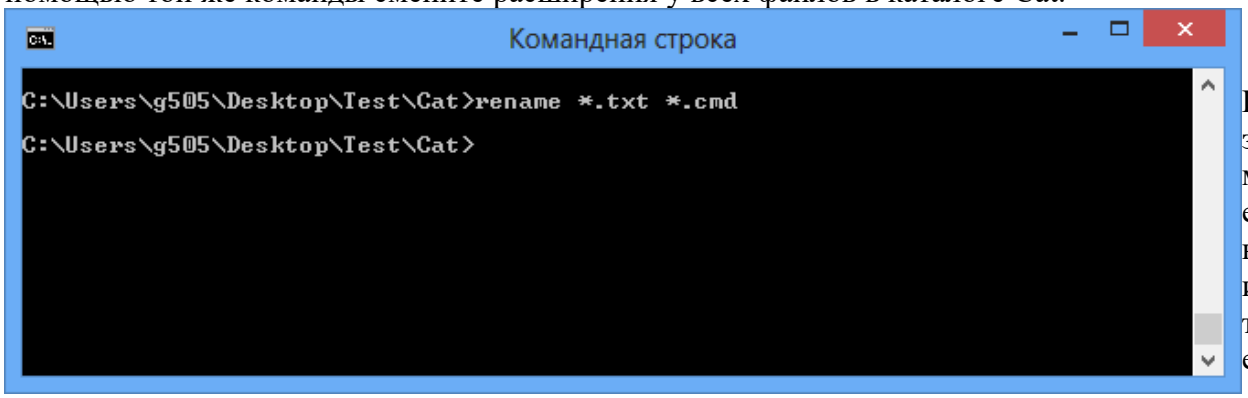
11. С помощью команды move переместите файл oneFile.txt в папку Test.



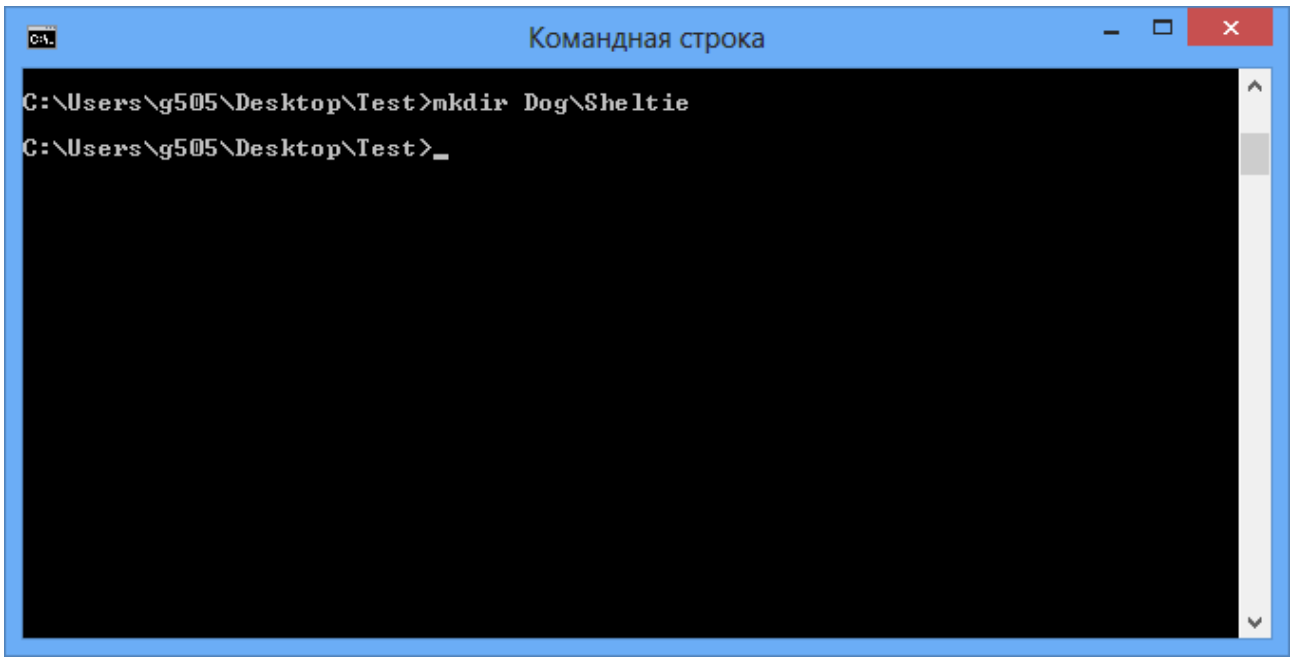
12. С помощью команды rename переименуйте файл oneFile.txt в newFile.txt.



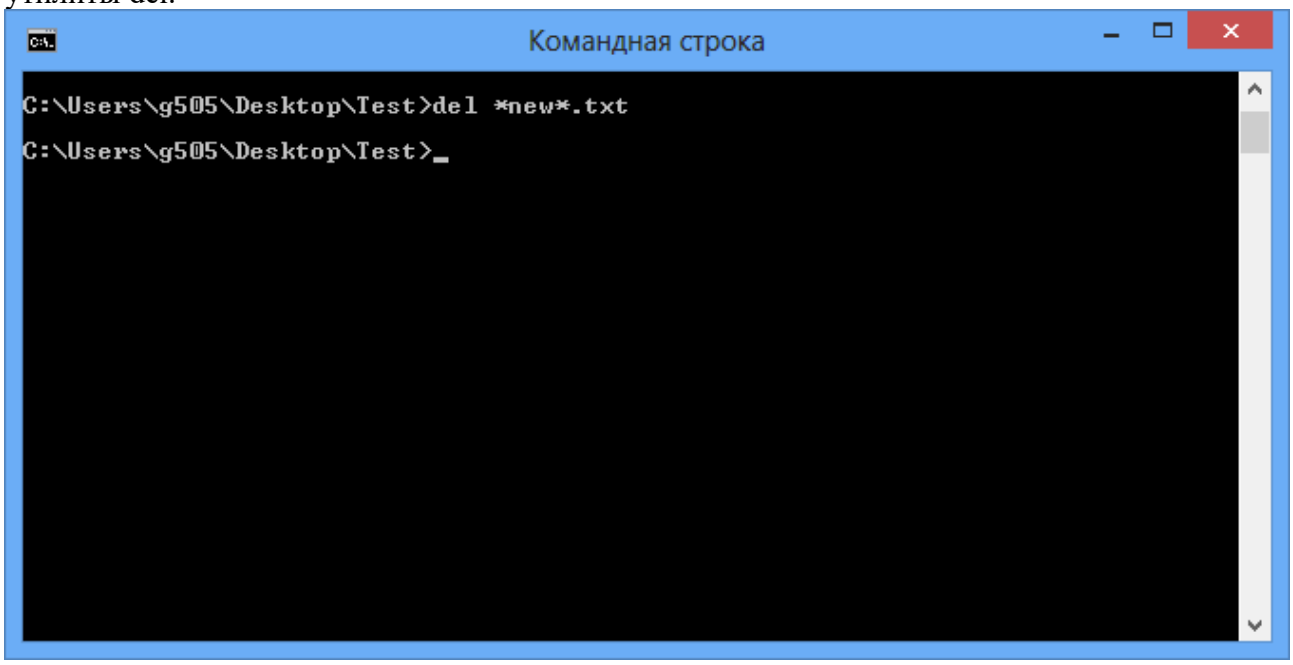
13. С помощью той же команды смените расширения у всех файлов в каталоге Cat.



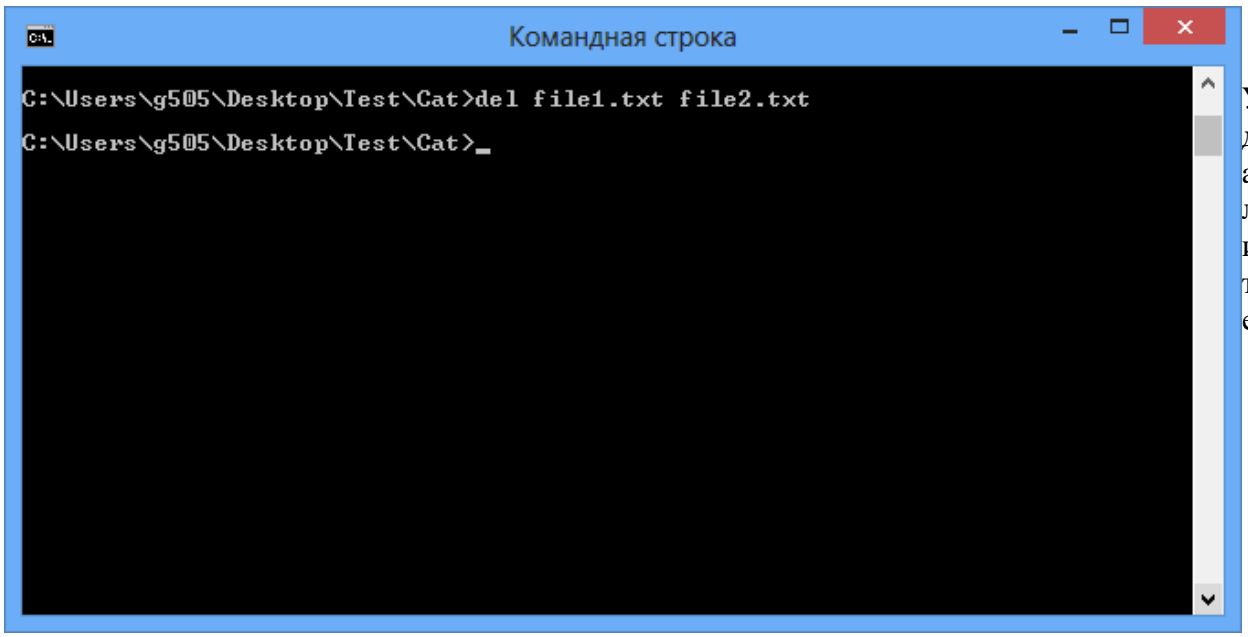
14. Измените расширение файлов на исходное.
15. Создайте в папке Test две цепочки вложенных каталогов с помощью команды mkdir:
    а) Каталог Dog, содержащий каталоги Bulldog и Sheltie.
    б) Каталог Bird, содержащий каталоги Flying и Non-flying. Каталог Non-flying содержит два подкаталога: Ostrich и Penguin.



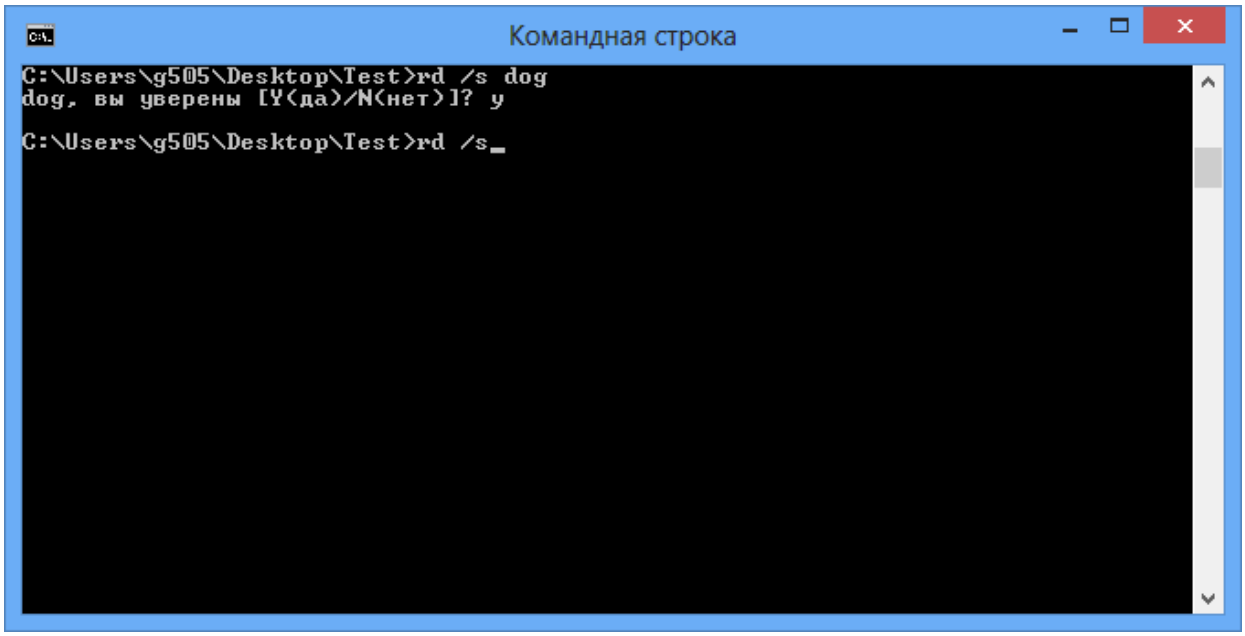
16. Используя команду tree . постройте дерево каталогов папки Test.
17. Удалите все файлы, содержащие в названии слово new, из каталога Test посредством утилиты del.



18. Удалите все файлы из каталога Cat.

19. 

Удалите каталоги Bird и Dog с помощью команды rd /s.



Практическая работа №3 «Сравнительный анализ офисных пакетов»

Цель: научиться выполнять сравнительный анализ офисных пакетов.

Форма отчета:
−выполнить задание;
−показать преподавателю;
−ответить на вопросы преподавателя.
Время выполнения: 2 ч

Первая версия OpenOffice представляла собой весьма неплохую альтернативу Microsoft Office. Однако в ней было достаточно большое количество недоработок, а также довольно ограниченная функциональность относительно решения от Microsoft. В связи с этим он не получил соответствующего возможностям распространения. Тем не менее, разработки

продолжались далее, и примерно во время выхода версии 1.1 стала доступна альфа-версия OpenOffice второй редакции, или 2.0.

OpenOffice, по аналогии с Microsoft Office, состоит из нескольких программ, которые и составляют пакет. Тем не менее, в то время как продукт от Microsoft поставляется во множестве модификаций (Standart, Professional, Enterprise и так далее), OpenOffice доступен только в одной версии. Первый является платным решением, и далеко не всем пользователям нужны все те функции, которые он предлагает. По этой причине и сделано такое разделение, чтобы пользователь сам мог выбрать, за какие функции он согласен платить.

Самыми известными приложениями Microsoft Office являются следующие:

· Word (текстовый редактор);
· Excel (электронные таблицы);
· Access (база данных);
· PowerPoint (электронные презентации);
· Outlook (почтовый клиент, органайзер).

Именно они и продублированы в OpenOffice. Естественно, их названия изменены:

· Writer (текстовый процессор)
Calc (электронные таблицы);
· Base (база данных);
· Impress (электронные презентации).

Аналога в OpenOffice нет только у Outlook. Вместе с рассматриваемым пакетом программ поставляются приложения под названием Draw и Math. Предназначение первого - это создание изображений, а второго - различных формул. По большому счёту, Draw содержит в себе все те функции, которые равномерно распределены в других компонентах OpenOffice. Например, здесь можно нарисовать какие-либо объекты, используя инструменты векторной графики, а также сделать диаграммы.

Что касается Math, то аналогичный инструмент из Microsoft Office носит название Microsoft Equation. Вкратце его функционал мы рассмотрим позже во второй статье вместе с обзором Writer (Math будет наиболее полезен как раз в качестве приложения к текстовому редактору).

Теперь остановимся на системных требования и моментах установки OpenOffice и Microsoft office. Оба пакета максимально комфортно работают на операционной системе Windows 2000 года и выше. Установятся и на менее мощную систему, однако скорость работы и обработки данных будет занимать длительное время, да и от сбоев никто не застрахован. А установка стандартная. Фактически однотипные окна с последовательными нажатиями «Далее» и «Готово», с выборами установки приложений и вызовом программы двойным щелчком. В требованиях и установке отличия OpenOffice от Microsoft office минимальны.

Главные особенности пакетов для пользователя начинаются с их скорости работы. Здесь МО нет равных. Документ МО открывается в 3 раза быстрее, чем документ ОО. Причем при сравнении документов незначительного объема содержащих текст. В пределах своего пакета открытие файлов примерно

одинаково. А связано это с тем, что ОС Windows не является родной для ОО, а потому ей каждый раз при вызове необходимо подгружаться в систему и библиотеки, а это занимает время. Это еще одно сравнение OpenOffice и Microsoft Office, которое влияет на оперативность работы.

К одной из особенностей OpenOffice относится наличие глобальных настроек, которые действительны и одинаковы для всех приложений пакета. Они могут быть заданы в любой из шести программ. Все настройки сгруппированы в виде структуры «дерево».

Теперь несколько слов о совместимости OpenOffice и Microsoft office, ведь при работе с офисными пакетами важно, чтобы документы, созданные в том или ином пакете, могли без сбоев открываться в разных. То, что файл doc корректно прочитается любой версией МО это ясно, аналогично и с ОО. Но если создавать файл в ОО, а открывать его в МО, возникнут недоразумения. С таблицами или редактированием текста в ОО придется потрудиться, многие функции несовершенны, требуют дополнительного пользовательского вмешательства. И затем то, что создано в ОО в МО открывается со сбоями. Тогда как документ МО в ОО открывается без изменений. Дело в том, что МО обладает более мощным функционалом и возможностями, в этом пакете множество замысловатых и простейших функций, которые позволяют максимально упрощать работу с пакетом. В ОО большинство из них тоже есть, однако их необходимо все время настраивать.

Интерфейс главного приложения OpenOffice.org - Writer - очень близок к Word. Та же строка меню, знакомые панели инструментов. В Writer нашлось место для большого количества полезных пиктограмм. В MS Word по умолчанию на панели инструментов выведено гораздо меньше нужных элементов. Writer же сразу подготовлен к комфортной работе. Большая часть экрана отведена под виртуальный лист, на котором и происходит работа с текстом.

В новых версиях OpenOffice.org лист расположен посреди экрана. Раньше лист был расположен с левого края, чем очень мешал некоторым пользователям. Вокруг листа расположена традиционная линейка, на которой устанавливаются величины отступов, выступов и ширина полей. Для начала стоит упомянуть возможность Writer сохранять документы в формате PDF. Раньше OpenOffice.org не отличался примерной работой этой функции, часто возникали проблемы с картинками. Теперь PDF-компонент работает на отлично. Пользователь даже может устанавливать степень сжатия изображения. Для удобства на панели инструментов есть кнопка «Экспорт в PDF». В MS Word, как известно, такой возможности нет.

Writer позволяет вставлять в текст изображения. Есть стандартная галерея, намного более скромная, чем в детище от Microsoft, зато с менее знакомыми картинками. Среди прочих можно выделить большое количество маркеров, разделяющих линий и пр. Можно вставить изображение из файла. Работа с графикой поставлена на порядок лучше, чем в Word. Картинку легко двигать в любых направлениях, менять размер, брать в рамочку, прикреплять к ссылке, кадрировать и делать многое другое. Окно настройки изображения удивляет количеством опций.

Кроме готового изображения, несложно создать самостоятельно.

Например, можно делать красочные надписи с помощью инструмента FontWork (аналог WordArt). А можно создать изображение при помощи панели рисования и автофигур. Делать это удобнее, чем в Word. Если же возможностей для рисования покажется мало, стоит воспользоваться другой программой из пакета OpenOffice.org - Draw. Ещё одна особенность Writer - по умолчанию ставится большое тире, что экономит время.

Несколько слов о проверке орфографии. «Заводской» Writer не понимает русский язык - требуется установка словарей. Для OOo 3.0.0 достаточно скачать модуль расширения и запустить. Не нужно даже закрывать Writer - словарь мгновенно устанавливается и тут же готов к работе.

Проблема OpenOffice.org Writer - подсчёт статистики. Word может выводить информацию о количестве страниц, знаков с пробелами/без, абзацев, слов и строк. Writer под силу только подсчитать количество слов и символов с пробелами. Не очень удобна работа с таблицами. Нельзя, как в Word, нарисовать таблицу карандашом. Организация последующего процесса заполнения и редактирования также далека от идеала.

OpenOffice Calc тоже похож на Excel. Разработчики лишь не заимствовали у Excel курсор мыши в виде большого плюса. Других особенностей интерфейс программы не несет. Этого же нельзя сказать про функционал. К его плюсам относится фильтр удаления. Если выделить часть таблицы и нажать кнопку Del, появится диалоговое окно, в котором можно выбрать, что именно удалить - математические формулы, объекты, примечания, числа, столбцы, строки и т.д. В плюсы можно записать и большее количество доступных цветов для заливки ячеек.

Другое отличие - большое количество встроенных математических формул. Правда, их названия записаны на английском языке.

Аналогов популярной программы Microsoft PowerPoint немного. Есть несколько бесплатных версий, которые способны воспроизводить готовые презентации. Но редактировать и создавать их с нуля под силу только OpenOffice.org Impress. Он совместим со стандартными презентациями *.ppt, может как открывать такой формат, так и сохранять данные в нем.

Трудно найти отличия как в интерфейсе, так и в функционале. Разве что готовых шаблонов презентаций в Impress на порядок меньше. Эти шаблоны не отличаются особым дизайнерским, но работать с ними несколько удобнее, особенно начинающему пользователю - Impress делает много подсказок. Довольно удобная функция - вставка в презентацию «плавающих» дату и время. На слайдах отводится специальное поле. Достаточно вписать один раз новые дату и время, и они появятся во всей презентации. Это может быть особенно полезным, когда презентацию переносят - не придется вносить изменения в каждый слайд.

Impress имеет и серьезные недостатки. К таковым можно отнести огромную ресурсоемкость. Воспроизведение сложной презентации может привести компьютер к зависанию. Программа может занимать до 500 Мб оперативной памяти. При этом совместимость презентаций хорошая, за редкими исключениями. В Impress иногда можно столкнуться с проблемой «рассогласования» страниц слайда и их миниатюр, расположенных слева.

OpenOffice.org Draw представляется как графический редактор. Правда,

надо сделать оговорку - это редактор деловой графики. Интерфейс его можно охарактеризовать как MS Word с расширенной панелью «Рисование», лишенный возможности работы с текстом. Кому-то может показаться, что интерфейс напоминает CorelDraw.

Неплохо поставлена работа с примитивными 3D-объектами. Автофигурам на панели «Рисование», можно придать объем. Причем не просто толщину - из плоского объекта можно сделать тело вращения. С получившимися фигурами можно совершать разные действия: менять размер и пропорции, накладывать свет и тень, выравнивать и пр.

OpenOffice.org Base позволяет создавать, редактировать или просто дополнять базы данных и не имеет существенных отличий от MS Office Access.

Задание

Составить 10 пунктов сравнительной характеристики по примеру Windows и linux.

Практическая работа №4 «Сравнительный анализ браузеров»

Цель: научиться выполнять сравнительный анализ браузеров.

Форма отчета:

−выполнить задание;

−показать преподавателю;

−ответить на вопросы преподавателя.

Время выполнения: 2 ч

Сравнительный анализ браузеров

Начнем с того, что браузер – это программа для просмотра веб-сайтов, выполняющая обработку, вывод веб-страниц и переход от одной веб-страницы к другой. В данной работе рассмотрим и проанализируем работу четырех веб-браузеров, а именно Internet Explorer, Google Chrome, Mozilla Firefox и Safari.

В основном подавляющее большинство популярных современных браузеров распространяются бесплатно или совместно с другими программными приложениями.

Например браузер Internet Explorer поставляется совместно с операционной системой Microsoft Windows. И является самым широко используемым браузером в мире. Internet Explorer имеет вкладки, блокировщик всплывающих окон, фишинг-фильтр, встроенный RSS-агрегатор, поддержку интернациональных доменных имен, средств групповой политики и возможность автообновления через Windows Update. Плюсами данного браузера является то, что в проводнике Windows стоит лишь прописать URL какого либо сайта и автоматически этой задачей займется IE и соответственно откроется этот сайт прописанный в строке Проводника. Обладает удобным и дружественным интерфейсом, привычным для всех пользователей Windows, что является большим преимуществом перед конкурентами.

Браузер Mozilla Firefox распространяется абсолютно бесплатно на некоммерческой основе, и поставляется совместно с многими дистрибутивами операционной системы Linux, например с Ubuntu. Mozilla — это не только браузер, это полноценный набор веб-приложений, но и набор приложений для работы с почтой, новостями, чат клиентом. Помимо этого в браузере присутствует дополнение, которое срабатывает при аварийном закрытии браузера, после запуска, браузер предложит восстановить последнюю сессию (с

теми вкладками, которые были полностью). По мнению калифорнийской компании Sauce Labs — разработчика платформы для тестирования приложений — в Firefox ошибки возникают реже, чем в других браузерах. Коэффициент ошибок в Safari — 0,15 %, Google Chrome— 0,12 %, и Firefox — 0,11 %. Таким образом, Firefox показал лучший результат — процент сбоев оказался наименьшим среди самых популярных веб-браузеров.

Браузер Safari распространяется совместно с операционной системой OS X и входит в состав операционной системы любого планшета корпорации Apple (iPad, iPad mini, iPad Air). Также этот браузер абсолютно бесплатен для операционной системы Microsoft Windows. Safari автоматически распознает веб-сайты, использующие нестандартные шрифты, и загружает их по мере необходимости. В Safari присутствует занятная функция – «Частный просмотр». Это режим, при котором не ведется история посещений, cookie не принимаются, пароли и вводимые данные не запоминаются. В Safari есть режим Reader удобного чтения, что является очень полезным для использования на мобильных устройствах.

Браузер Google Chrome активно распространяется в интернете, благодаря удачной рекламной компании и совместимости браузера с операционными системами Windows, Linux, OS X. У этого браузера достаточно приятно оформлена панель вкладок, панель закладок и строки ввода, состояния и ссылки. Присутствуют кнопки «Вперед», «Назад», «Обновить». Также на странице имеется многофункциональный поиск, выделение найденных сочетаний. Присутствует перевод страницы на определенный язык. Достаточно функциональное сохранение пользовательской информации. При установке всегда запрашивается логин и пароль от сервиса GOOGLE. Вся информация, подобно закладкам, истории и информации о загрузках хранится на серверах компании Google. Google Chrome направлен на повышение безопасности, стабильности и удобства.

Рассмотрим распространенность этих браузеров с учетом различных технических устройств (персональные компьютеры, ноутбуки, нетбуки, планшеты, смартфоны). Согласно исследованиям NetMarketShare в мировом сегменте на апрель 2014 года пользователи персональных компьютеров, ноутбуков и нетбуков в большинстве своем предпочитают пользоваться браузером Internet Explorer, количество пользователей которого на этих устройствах составляет 57,88 %, следующий по показателям идет Google Chrome с показателем 17,92 %, потом Firefox с 17,00 % и Safari с 5, 66%, Opera – 1,14 %, остальные – 0, 39%. Из этого сделаем вывод что на персональных компьютерах, ноутбуках и нетбуках господствуют три ключевых браузера – Internet Explorer с подавляющим большинством, и Google Chrome на пару с Mozilla Firefox. Это вполне закономерна, учитывая повсеместное распространение Internet Explorer в комплекте с любой операционной системой семейства Windows. А также отличной маркетинговой компании Google, позволяющей подняться Chrome в рейтинге потребителей. А вот заслуги Firefox исключительно в гибкости и разнообразия плагинов, тем оформления и прочим расширениям, делающих браузер более привлекательным в глазах потребителей.

Теперь перейдем к распространенности браузеров на планшетах и смартфонах. Согласно исследованиям NetMarketShare в мировом сегменте на апрель 2014 года пользователи мобильных устройств, планшетов и смартфонов в большинстве своем предпочитают пользоваться браузером Safari, количество пользователей которого на мобильных устройствах составляет 51,77%. Следующий по показателям идет Android Browser с показателем 22,56%. Следом Chrome с 14,4%. А у Opera Mini - 5,05%. Internet Explorer – 2,26%, у BlackBerry – 0,52%. Остальные – 3,45%. Из этого делаем вывод, что на мобильных устройствах, таких как планшеты и смартфоны в мировом сегменте за 2014 год преобладает браузер Safari. Браузеры Android Browser и Chrome занимают второе и третье место на мировой арене.

Характеристики быстродействия браузеров:

Инициаторы большинства сравнений популярных веб-браузеров почему-то не учитывают тот факт, что нынешнее поколение таких продуктов потребляет ресурсы гораздо интенсивнее, чем предыдущие. Новые операционные системы, например Windows

8, становятся все более «изящными», размещаясь на маломощных мобильных устройствах (планшетных компьютерах, первом поколении ультрабуков) или на тонких клиентах. Однако браузеры, являющиеся, пожалуй, самыми важными приложениями любого ПК, становятся все более прожорливыми. Компания Acid3 предоставляет нам возможность протестировать работу CSS и JavaScript у популярных браузеров. Одновременное открытие в браузере сразу 15 вкладок — сейчас не такое уж редкое событие. Независимо от того, как вы это оцениваете, браузер является одним из самых крупных и интенсивных потребителей компьютерных ресурсов. Почему?

Прежде всего потому, что веб-браузер – это не программа для просмотра документов, а веб-сайт – не простой документ. Браузеры представляют собой сложную и развитую среду программирования. Технологии HTML5, JavaScript и Flash гораздо мощнее и запутаннее любого документа.

В официальных стабильных версиях Internet Explorer и Chrome поглощали много памяти в сценарии с 15 вкладками. (Думаю, что если число вкладок довести до 40 и более, объем потребляемой памяти вырастет еще существеннее.) Превзошел же всех Firefox. Он использовал всего половину объема памяти, понадобившейся другим браузерам в этом сценарии. Если при использовании базовых ресурсов объем занимаемой браузером памяти находился на уровне 68 Мбайт, то после открытия пяти вкладок он увеличился до 218 Мбайт.

Какое влияние это окажет на производительность в реальных условиях? Internet Explorer и Chrome выглядели весьма инертными. Переключение между вкладками осуществлялось с заметными задержками. Общая производительность системы и скорость реакции также упали. У Firefox видимой разницы в производительности не обнаруживается.

Возможности расширения функционала браузеров:

Расширения добавляют новые возможности в браузер или разрешают модифицировать существующие настройки. Они могут добавить практически что угодно: от кнопки на панели инструментов до совершенно новых возможностей.

Рассмотрим расширения в браузере Mozilla Firefox. Механизм расширений превращает изначальную аскетичность браузера Mozilla Firefox в одно из основных преимуществ: устанавливая расширения, пользователь может выбрать именно ту функциональность, которая необходима ему для комфортного сёрфинга, при этом не занимая рабочее пространство и ресурсы ненужными функциями.

Необходимо заметить, что *расширения* (например, Adblock Plus и Firebug), *темы* («обои» и «полные»), *локализации* и *лагины* (например, Adobe Flash, QuickTime, Java) к Firefox — не одно и то же, а лишь различные виды *дополнений*.

Есть несколько типов дополнений по разному настраивающих Firefox:

- добавляют новые функции в Firefox или изменяют существующую функциональность. Существуют расширения, которые позволяют блокировать рекламу, загружать видео с веб-сайтов, более тесно интегрироваться с социальными сетями и добавлять функции, которые вы видите в других приложениях.
- это легкие темы, которые используют фоновые изображения для настройки панелей инструментов вашего Firefox.
- **Инструменты поиска** добавляют дополнительные варианты в выпадающий список панели поиска. Эти инструменты позволяют вам быстро произвести поиск на любом веб-сайте.
- **Словари и локализации** добавляют поддержку дополнительных языков в Firefox.
- помогают Firefox отображать или понимать различные типы медиа, такие как Adobe Flash или Apple Quicktime.

В большинстве случаев дополнения можно установить, просто щёлкнув по кнопке установки. Дополнениями можно управлять, отключать или удалять через окно управления Дополнениями в Firefox.

Теперь рассмотрим расширения в браузере Chrome. Расширения Google Chrome позволяют расширить возможности и функции браузера Chrome. Та или иная функция

может быть полезна для некоторых людей, но не для всех. Расширения позволяют добавлять в Google Chrome только нужные возможности, избегая накопления функций, которые не используются. Google создал специальную галерею расширений от третьих лиц. Пользователи могут установить темы, изменяющие внешний вид браузера. Была создана галерея, которая включает в себя как темы от Google, так и темы от третьих лиц. Интернет магазин Chrome — онлайн интернет-магазин компании Google, позволяющий пользователям устанавливать и запускать веб-приложения, расширения и темы для браузера Google Chrome. Сюда также входят расширения, и темы. Магазин часто сравнивают с родственным проектом Android Market и App Store от корпорации Apple.

Безопасность – настройки и статистика обнаружения уязвимостей.

Вкладка «Безопасность» Internet Explorer позволяет задать и изменить параметры, которые могут защитить компьютер от потенциально опасного или вредоносного содержимого в сети.

Параметры безопасности в браузере Internet Explorer:

Откройте веб-обозреватель Internet Explorer.

Нажмите кнопку **Сервис** (ALT + F) и выберите пункт **Свойства браузера**

Перейдите на вкладку **Безопасность**.

Internet Explorer относит веб-узлы к одной из четырех зон безопасности: зона Интернета, зона интрасети, зона надежных узлов и зона ограниченных узлов. Зона, к которой относится данный веб-узел, задает его настройки безопасности.

Описашем четыре зоны безопасности Internet Explorer:

Уровень безопасности, заданный **для зоны Интернета**, применяется ко всем веб-узлам по умолчанию. Заданный уровень безопасности для данной зоны - умеренно высокий (однако его можно изменить на умеренный или высокий). Данный параметр безопасности не используется только для узлов в зоне местной интрасети или для специально отнесенных к зонам надежных или ограниченных узлов.

Уровень безопасности, заданный **для зоны местной интрасети**, применяется к веб-узлам и содержимому, которое хранится в сети организации или в рабочей сети. Заданный уровень безопасности для зоны местной интрасети - умеренный (однако его можно изменить на любой другой).

Уровень безопасности, заданный **для зоны надежных узлов**, применяется к веб-узлам, которые специально отнесены к этой зоне как не опасные для компьютера или данных. Заданный уровень безопасности для зоны надежных узлов - умеренный (однако его можно изменить на любой другой).

Уровень безопасности, заданный **для зоны ограниченных узлов**, применяется к веб-узлам, потенциально опасным для компьютера и данных. Отнесение веб-узлов к зоне ограниченных узлов не блокирует их, но запрещает выполнение сценариев и любого активного содержимого. Заданный уровень безопасности для зоны ограниченных узлов - высокий и изменению не подлежит.

Настройки безопасности в браузере **Mozilla Firefox** можно регулировать в меню «Настройки» во кладке «Зашита». Панель «Защита» содержит настройки, связанные с обеспечением безопасности при работе в Интернете. Рассмотрим некоторые пункты в этом окне:

Предупреждать при попытке веб-сайтов установить дополнения: Firefox всегда будет выдавать запросы на подтверждение установки дополнений. Чтобы предотвратить выдачу сайтами незапрошенных вами запросов на установку, которые могут привести к случайной установке дополнения, Firefox будет предупреждать вас при попытке веб-сайта установить дополнение и блокировать запрос на установку.

Блокировать веб-сайты, подозреваемые в атаках: Установите этот флажок, если хотите, чтобы Firefox проверял, не является ли сайт, который вы посещаете, сайтом, мешающим нормальному функционированию компьютера или отсылающим ваши персональные данные сторонним лицам через Интернет.

Блокировать веб-сайты, подозреваемые в мошенничестве: Установите этот флажок, если вы хотите, чтобы Firefox производил проверку, не пытается ли посещаемый

вами сайт ввести вас в заблуждение и выманить у вас персональные данные (этот вид мошенничества часто называется *фишингом*).

Задание

Составить 10 пунктов сравнительной характеристики по примеру двух любых браузеров.

Практическая работа № 5«Сравнительный анализ средств просмотра видео»

Цель: научиться выполнять сравнительный анализ средств просмотра видео.

Форма отчета:
−выполнить задание;
−показать преподавателю;
−ответить на вопросы преподавателя.
Время выполнения: 2 ч

В последнее время появилось огромное количество различного софта для смартфонов. Большинство предоставляемого софта является платным. Следует заметить, что для смартфонов с операционной серией S60, вообще, бесплатного софта очень немного. Пользователю, конечно же, очень хочется, чтобы софт, который он выбрал, работал адекватно, без "глюков". Но на данный момент очень немного программ может похвастаться своей безупречной работой - скажем, Вы установили софт, Вам он показался удачным и замечательным, но в процессе углубленной работы начинают всплывать "подводные камни". Исходя из этих соображений, мы с Вами проведем сравнительный обзор четырех программ предназначенных для просмотра видеофайлов (DivX Player, MobiFactor PowerMovie, DVDPlayer, SmartMovie), и определим их положительные и отрицательные черты работы.

Первое на что следует обратить внимание - это на размер дистрибутива:

- DivX Player - 952,6 Kb
- MobiFactor PowerMovie - 440,4 Kb
- DVDPlayer - 243,9 Kb
- SmartMovie - 1,9 Mb

Как видим, в размере дистрибутива, бесспорно, выигрывает программаDVDPlayer. Интерфейс у программ DivX Player, MobiFactor PowerMovie, SmartMovie англоязычный, а у программы DVDPlayer русскоязычный. Из рассматриваемых нами программ лишь DivX Player распространяется бесплатно. Ну что ж, заканчиваем наше вступление, и переходим к поэтапному сравнительному анализу, начиная от запуска утилиты, и заканчивая подробным описанием меню.

1) Запуск:

В программе DivX Player при запуске утилиты не появляется никаких заставок и предложений о регистрации. Данный факт свидетельствует о том, что софт распространяется бесплатно, и разработчики не делали своей основной целью "поднятие" денег на разработке данной утилиты.

В программе MobiFactor PowerMovie при запуске утилиты нам предлагается сделать выбор - работать с Trial версией программы, либо провести незамедлительную регистрацию утилиты.

В программе DVDPlayer при запуске утилиты появляется требование о вводе регистрационного кода, и нам сообщается о том, что в незарегистрированной версии программы мы можем просматривать видео лишь на протяжении 3 минут:

В программе SmartMovie при запуске утилиты не появляется никаких заставок и предложений о регистрации.

2) Интерфейс и структура программ:

В данном пункте обзора мы рассмотрим общее сравнительное описание интерфейсов сравниваемых программ:

DivX Player - интерфейс главного меню программы представлен пользователю списком аудиофайлов. В данном случае программой предоставляется две, так называемые, дэмки для проведения первичного просмотра данного плеера. Огорчил тот факт, что программа, по всей видимости, не сканирует память смартфона на предмет обнаружения *.avi файлов, и следственно не заносит их в список воспроизведения. Поэтому, чтобы видео файл очутился в списке воспроизведения нам необходимо поместить его в папку "videos", которая появляется после установки программы. Хотелось бы также заметить, что внизу главного меню программы присутствуют два пункта: слева пункт Options (содержит в себе все необходимые функции для адекватной работы с утилитой), справа находится пункт Exit (с помощью данного пункта мы можем выйти из программы в главное меню нашего смартфона):

MobiFactor PowerMovie - интерфейс главного меню программы представлен пользователю списком видео файлов, обнаруженных программой на дисках памяти нашего смартфона. Должен заметить, что сканирование памяти смартфона программа проводит автоматически в процессе её запуска. Навигацию по списку, как и запуск проигрывания видео, мы можем осуществлять с помощью джойстика нашего смартфона. При выборе того, или иного видео файла появляется меню проигрывателя. Расположение меню проигрывателя на экране смартфона мы можем настраивать самостоятельно. Как пример, на верхнем скриншоте я привел меню проигрывателя с левосторонней ориентацией. Систематику изменения расположения меню проигрывателя на экране смартфона, мы рассмотрим чуть позже. А пока я бы хотел обратить внимание на то, что в меню проигрывания присутствует дорожка проигрывания записи, клавиша пуск/стоп и возможность регулировки звука. Как видите ничего лишнего - все просто, и не предвзято. Внизу главного меню программы присутствуют два пункта: слева пункт Функции (содержит в себе все необходимые функции для адекватной работы с утилитой), справа находится пункт Назад (с помощью данного пункта мы можем выйти из программы в главное меню нашего смартфона):

DVDPlayer - интерфейс главного меню программы очень прост, и представлен нам списком всех видео файлов, которые программа нашла на дисках памяти нашего смартфона. Проигрывание того, или иного видео файла мы можем начать, нажав джойстиком на его названии. К сожалению, скриншот меню проигрывания сделать не удалось, из-за недостатков программы захвата изображения. Могу лишь заметить, что меню проигрывания очень простое - все, что мы можем сделать, это лишь отрегулировать звук и осуществить быструю перемотку просматриваемого видео. Все эти действия мы осуществляем с помощью джойстика нашего смартфона. Внизу главного меню программы присутствуют два пункта: слева пункт Опции (содержит в себе все необходимые функции для адекватной работы с утилитой), справа пункт Воспр. (позволяет запустить процесс воспроизведения видео файлов):

SmartMovie - интерфейс главного меню программы представлен пользователю списком обнаруженных на нашем смартфоне видеофайлов. Навигативное перемещение по списку мы можем выполнять с помощью джойстика нашего смартфона. Внизу главного меню программы

присутствуют два пункта: слева пункт Menu (содержит в себе необходимые функции для оптимизации работы с утилитой), справа находится пункт Exit (с помощью данного пункта мы можем выйти из программы в главное меню нашего смартфона):

3) Подробное описание Меню:

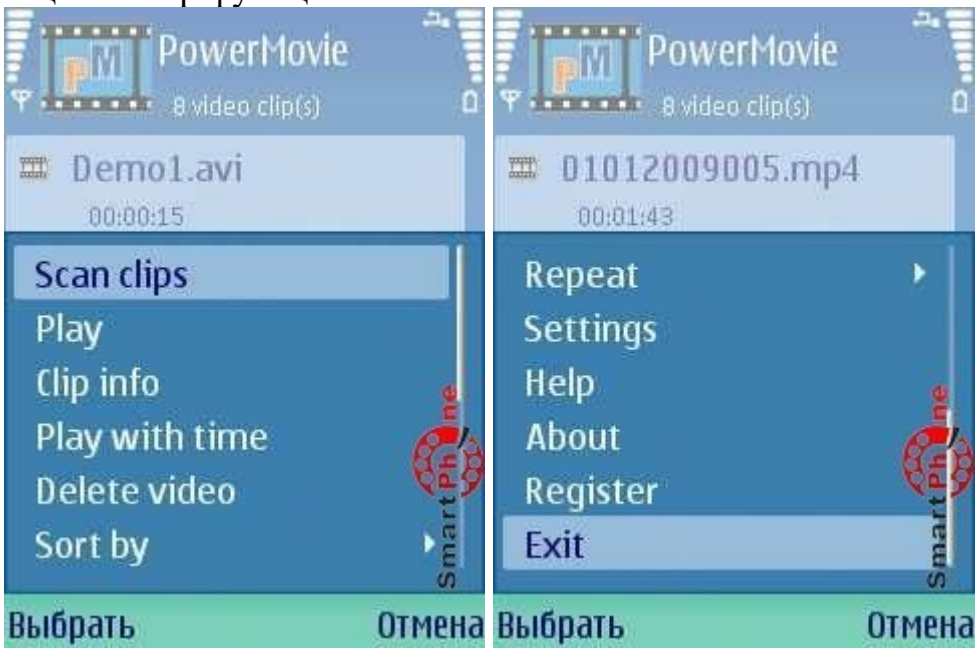
После рассмотрения общего интерфейса главного меню программы перейдем к сравнительному рассмотрению пунктов, содержащих необходимый набор функций для адекватной работы программ:

Начнем с DivX Player - в данной программе функционально содержащий пункт называется Options, и содержит следующий набор функций:

С помощью пункта Options, который представлен выше на скриншоте, мы можем: воспроизводить тот, или иной видеофайл из списка главного меню программы; удалять ненужные нам *.avi файлы из списка главного меню программы; обновлять список видеофайлов, в том случае если скопированный нами видеофайл в папку "videos", не отобразился в списке воспроизведения, главного меню программы; сортировать видеофайлы из

списка воспроизведения; работать с основными настройками программы; воспользоваться справочными данными программы; выйти в сеть интернет и автоматически перейти на сайт разработчика. Лично у меня, при выборе этой функции программа долго думала, а потом выдала ошибку подключения; зарегистрировать данный софт, как я понял, данная функция чисто для проформы; выйти из программы в главное меню нашего смартфона.

Перейдем к утилите MobiFactor PowerMovie - в данной программе функционально содержащий пункт называется Функции, и содержит следующий набор функций:



С помощью пункта Функции, который представлен выше на скриншоте, мы можем: возобновлять операцию поиска видео файлов на дисках памяти нашего смартфона, и занесения их, в случае обнаружения, в список воспроизведения программы; запустить проигрывание того, или иного видео, из списка главного меню программы; просматривать достаточно полную информацию о видео файлах, из списка главного меню программы; задавать время, с которого следует начинать проигрывание той или иной видеозаписи; удалять ненужные нам видео файлы из списка главного меню программы; сортировать видео из списка главного меню программы по времени или продолжительности; задавать параметры автоматического повторения проигрывания видеозаписей; работать с основными настройками программы; воспользоваться справочными данными программы; просмотреть информацию о данной версии программы и её разработчиках; произвести регистрацию данной версии программы, путем ввода регистрационного кода, полученного при покупке софта; выйти из программы в главное меню смартфона.

Перейдем к утилите DVDPlayer - в данной программе функционально содержащий пункт называется Опции, и содержит следующий набор функций:

С помощью пункта Опции, который представлен выше на скриншоте, мы можем: задавать время, с которого следует начинать проигрывание той или иной видеозаписи; возобновлять операцию поиска видео файлов на дисках памяти нашего смартфона, и занесения их, в случае обнаружения, в список воспроизведения программы; работать с справочными данными программы, так любезно предоставленными нам разработчиками софта; просмотреть информацию о данной версии программы и её разработчиках; выйти из программы в главное меню смартфона.

Перейдем к утилите SmartMovie - в данной программе функционально содержащий пункт называется Menu, и содержит следующий набор функций:

С помощью пункта Menu, который представлен выше на скриншоте, мы можем: апускать проигрывание того, или иного видео из списка главного меню программы; просматривать детальную информацию о видеофайлах из списка главного меню программы; удалять ненужные видеофайлы из списка главного меню программы; переименовывать видеофайлы из списка главного меню программы; сортировать видеофайлы из списка главного меню

программы; работать с основными настройками утилиты, искать обновление данной версии утилиты в сети интернет, а также запускать сканирование дисков памяти нашего смартфона на предмет обнаружения новых видеофайлов; просмотреть информацию о используемой нами утилите и её разработчиках; провести регистрацию данной версии программы; выйти из программы в главное меню смартфона.

Подведем итоги:

В подведении итогов я бы хотел привести отдельно плюсы и минусы каждого рассмотренного выше софта, и, исходя из этого, сделать общий вывод:

- Программа DivX Player:

(+): К плюсам можно отнести достаточно широкий спектр настроек утилиты; данный софт распространяется бесплатно.

(-): *К минусам можно отнести слишком отсутствие русского языка интерфейса; программа не сканирует память смартфона на предмет обнаружения *.avi файлов, и следственно не заносит их в список воспроизведения. Заносить видеофайлы в список нужно вручную; присутствуют несколько настроек, о назначении которых можно только догадываться.*

- Программа MobiFactor PowerMovie:

(+): К плюсам можно отнести поддержку популярных форматов: AVI(DivX, XviD), MP4; поддержку трех разных ориентаций экрана; достаточно высокое качество воспроизводимого видео; возможность начала просмотра с заданного времени; возможность сортировки видео из списка главного меню программы, по времени или продолжительности; достаточно широкие настройки программы.

(-): *В процессе пользования программой минусов замечено не было.*

- Программа DVDPlayer:

(+): К плюсам можно отнести простой русскоязычный интерфейс программы; возможность полноэкранного просмотра видео файлов; возможность начала просмотра с назначенного пользователем времени; автоматическое отображение всех имеющихся на телефоне Avi файлов; возможность быстрой перемотки просматриваемого видео.

(-): *К минусам можно отнести разве что слишком высокую цену утилиты, которая составляет 14,95 USD.*

- Программа SmartMovie:

(+): К плюсам можно отнести интуитивно простой интерфейс программы; возможность просмотра, как портретного, так и пейзажного видео поддержку субтитров, можно смотреть видео на многих языках; поддержку кодеков DirectShow; бикубическую интерполяцию - максимальное качество сжатого видео.

(-): *В процессе пользования программой минусов замечено не было.*

Вывод: *из рассмотренных выше программ, на мой взгляд, лучшей является программа SmartMovie, которая сможет в полной мере обеспечить достойный просмотр видео.*

Задание

Составить 10 пунктов сравнительной характеристики по примеру двух любых программ просмотра видео.

Практическое занятие №6 «Обратное проектирование алгоритма»

Цель работы: закрепить умения и навыки в разработке несложных алгоритмов и в построении их блок-схем

Форма отчета:
−выполнить задание;
−показать преподавателю;
−ответить на вопросы преподавателя.
Время выполнения: 2 ч

Оборудование, приборы, аппарату ра, материалы: персональный компьютер с выходом в Интернет.

Краткие теоретические сведения.
Алгоритм как информационная модель. Алгоритмы лежат в основе современных информационных технологий. Алгоритм является информационной моделью процесса решения задачи. Исполнитель алгоритма выполняет алгоритм формально, не вникая в содержание поставленной задачи. Человек при разработке и исполнении алгоритмов использует язык блок-схем. Блок- схема позволяет сделать алгоритм более наглядным и выделить в нем основные алгоритмические структуры (линейная, ветвление, цикл и др.). Человек может по блок-схеме легко проследить выполнение алгоритма, так как элементы блок-схем соединены стрелками, указывающими последовательность действий.

Вычислить высоты треугольника со сторонами a, b, c, пользуясь формулами:
h_a=
h_b =
h_c=
где p = (a + в + c) / 2.
Решение задачи состоит из трех этапов:
1) ввод исходных данных сторон треугольника a,b,c;
2) расчет по формулам;
3) вывод полученных результатов – высот треугольника h_a ,h_b, h_c .

У вас должна получиться следующая блок-схема.

Пример 1. Вычислить значение функции
$Z = x^3/y$, где $y = \sin(nx) + 0{,}5$.
Казалось бы, что решение этой задачи описывает алгоритм линейной структуры. Однако для удовлетворения свойства массовости и результативности алгоритма необходимо, чтобы при любых исходных данных был получен результат или сообщение о том, что задача не может быть решена при заданных данных. Действительно, если y = 0, то задача не может быть решена, так как деление на нуль невозможно. Поэтому в алгоритме необходимо предусмотреть условие y = 0 и выдать в качестве результата информацию «y = 0».
Таким образом, вычислительный процесс имеет две ветви. В одной ветви при y = 0 необходимо вычислить и вывести значение переменной z, в другой – вывести текст «y = 0». Такой вычислительный процесс можно описать следующей условной формулой:

Вычислить $z = x^3/y$, если $y \neq 0$
Z=
Вывести «y = 0» если, y = 0
У вас должна получиться следующая блок-схема.

Задание 3
Вычислите периметр и площадь прямоугольного треугольника, если известны катет и острый угол. Записать блок-схему алгоритма.
Контрольные вопросы

1. Какой алгоритм называется линейным?
2. Какой алгоритм называется циклическим?

3. Что представляет собой алгоритм ветвления?

Практическая работа № 7«Планирование code-review»

Цель: - усвоение принципов бюджетного планирования

Форма отчета:

−выполнить задание;

−показать преподавателю;

−ответить на вопросы преподавателя.

Время выполнения: 2 ч

Задание: - составить график движения материалов
- составить баланс материальных ресурсов
- составить график денежных поступлений и выплат
- составить баланс денежных средств

Данные для выполнения работы:

На 01.01. 2014г. на предприятии имеются следующие ресурсы:

- материальные запасы – 210 000р.
- денежные средства – 720 000р.

В течении первых трех месяцев года произошли следующие изменения:

Январь	**Февраль**	**Март**
– поступление выручки от реализации продукции 260 000р. - выплата зарплаты 80 000р. - налоги 30 000р. - оплата материалов – 80 000 р. - отпуск материалов на производство – 60 000р. - продажа материалов – 10 000р.	- поступление выручки от реализации продукции 270 000р. - выплата зарплаты 85 000р. - налоги 31 000р. - оплата материалов – 88 000р. - отпуск материалов на производство – 64 000р. - продажа материалов 10 000р.	- поступление выручки от реализации продукции 280 000р. - выплата зарплаты 85 000р. - налоги 31 000р. - оплата материалов – 96 000р. - отпуск материалов на производство – 69 000р.

<u>В дальнейшем планируется ежемесячное:</u>

- увеличение выручки на 10 000р.
- увеличение налогов на 1000р.
- увеличение зарплаты на 3 000р.
- увеличение отпуска материалов на производство на 5 000р.
- увеличение затрат на оплату материалов на 8 000 р.

<u>Помимо этого:</u>

- в апреле планируется поступление от продажи основного средства – 110 000р.
- в мае реализация материалов на 40 000р.

График движения материалов

месяц	Остаток на начало месяца	Поступление материалов от поставщика	Отпуск материалов на производство	Продажа материалов	Остаток на конец мес.
01					
02					
03					
04					
05					
06					
07					
08					

09					
10					
11					
12					
Итого:	-				-

Плановый баланс материальных ресурсов

Источники ресурсов	Сумма, руб.	Распределение ресурсов	Сумма, руб.
Остаток на начало года		Текущее потребление	
Поступление материалов от поставщика		Реализация на сторону	
Прочие поступления		Остаток на конец года	
Итого		Итого	
Баланс		Баланс	

График денежных поступлений и выплат

	Остаток на начало месяца	Поступле-ние выручки	Поступле-ние от реализации материалов	Поступле-ние от реализации основных средств	Выплата зарплаты	Уплата налогов	Оплата поставщикам за материалы	Остаток на конец мес.
01								
02								
03								
04								
05								
06								
07								
08								
09								
10								
11								
12								
Итого:	-							-

Плановый баланс денежных средств

Источники ресурсов	Сумма, руб.	Распределение ресурсов	Сумма, руб.
Остаток на начало года		Оплата поставщикам	
Поступление выручки		Прочие выплаты	
Прочие поступления		Остаток на конец года	
Итого		Итого	
Баланс		Баланс	

Практическая работа № 8«Проверки на стороне клиента»

Цель: - научиться проводить проверки на стороне клиента

Форма отчета:

−выполнить задание;

−показать преподавателю;

−ответить на вопросы преподавателя.

Время выполнения: 2 ч

Для пересылки своих параметров в JavaScript код атрибут должен поддерживать интерфейс *IClientValidatable*, определенный в пространстве имен System.Web.Mvc. При это необходимо реализовать всего лишь один метод *GetClientValidationRules()*, который возвращает перечень правил. Каждое правило включает список значений для настройки функции проверки и соответствующего сообщения об ошибке.

Откроем исходный код класса *EqualAttribute* и добавим поддержку указанного интерфейса. С помощью редактора Visual Studio создадим заготовку его метода и добавим в неё следующий код:

```
?
1
2       namespace MVCDemo.Attributes.Validation
3       {
4          using System;
5          using System.Collections.Generic;
6          using System.ComponentModel.DataAnnotations;
7          using System.Web.Mvc;
8
9          [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inheri
1          public class EqualAttribute : ValidationAttribute, IClientValidatable
0          {
1             private readonly object _valueToCompare;
1
1             .........
2
1             #region IClientValidatable Members
3
1             public IEnumerable<ModelClientValidationRule> GetClientValidation
4                ModelMetadata metadata, ControllerContext context)
1             {
5                var rule = new ModelClientValidationRule() {
1                   ErrorMessage = this.FormatErrorMessage(metadata.DisplayName
6                   ValidationType = "equal"
1                };
7
1                rule.ValidationParameters.Add("valuetocompare", this._valueToCo
8
1                yield return rule;
9             }
2
0             #endregion
2          }
       }
```

Теперь подробно рассмотрим реализацию метода *GetClientValidationRules()*.

Первый параметр данный метод содержит метаданные свойства, для которого назначен атрибут. В частности, из него можно получить отображаемое имя свойства, которое будет использовано для создания текста сообщения об ошибке. Вторым параметром является контекст Контроллера, включающий его данные и значения из исходного запроса.

Поскольку в данном случае только одно правило, то создается перечисление из единственного экземпляра *ModelClientValidationRule*. В нем сохраняются:

- *ErrorMessage* – сообщение об ошибке.
- *ValidationType* – уникальное имя данного правила (запомним его, т.к. оно не раз будет использовано в клиентском коде).
- *ValidationParameters* – коллекция пар "ключ-��значение", в которой сохраняются необходимые для передачи в JavaScript параметры (имена ключей также потребуются в коде на JavaScript).

Таким образом, в результате вызова данной реализации *GetClientValidationRules()* будет создано правило "equal", содержащее значение для сравнения "valuetocompare" и строку с текстом сообщения об ошибке.

Каркас ASP.NET MVC 3 сохранит эти значения в HTML теге соответствующего поля в следующем виде:

```
?
1       <input name="AgreementAccepted" class="check-box"
2       id="AgreementAccepted" type="checkbox"
3       data-val="true"
4       data-val-required="The I have read the EULA and I agree with it field is requ
5       data-val-equal-valuetocompare="True"
6       data-val-equal="You must agree with the EULA."
7       value="true"/>
```

Обратите внимание, имя правила используется для создания названий атрибутов HTML тегов. Сообщение об ошибке при этом располагается в "data-val-[имя-правила]", а значения параметров в "data-val-[имя-правила]-[ключ]". Кроме того, в именах правил и названиях ключей можно использовать только строчные буквы без пробелов и других символов.

Создаем JavaScript с алгоритмом проверки

Все готово для того, чтобы программа на JavaScript смогла получить данные и передать их в функцию проверки. Но перед тем как продолжить, давайте выберем место хранения для файлов кодом проверок на JavaScript. Для этого в папке Scripts создадим папку Validation. Кроме того, называть все

создаваемые файлы будем по аналогии с файлами, содержащими исходный код атрибутов.

Чтобы в очередной раз не изобретать колесо, ядро ASP.NET MVC 3 использует библиотеку jQuery для клиентской части веб-приложения. С её помощью реализован функционал проверки в браузере свойств, отмеченных стандартными атрибутами. Не будем углубляться в тонкости её работы и рассмотрим только необходимые для решаемой задачи детали.

Проверка данных на стороне клиента выполняется дополнением jQuery Validation plugin. Его работа сводится к следующему: при изменении поля необходимо вызвать функцию проверки значения и, в случае ошибки, вывести текст полученного сообщение на соответствующе место страницы. Данный механизм доступен через объект *jQuery.validator*. Используя его функцию *addMethod()* необходимо добавить новое правило проверки в их список.

В результате, в папке Scripts\Validation создадим файл EqualAttribute.js, содержащий следующий код:

```
?
1       /// <reference path="../jquery-1.4.4-vsdoc.js" />
2       /// <reference path="../jquery.validate-vsdoc.js" />
3       /// <reference path="../jquery.validate.unobtrusive.js" />
4
5       jQuery.validator.addMethod("equal", function (value, element, param) {
6           if ($(element).attr("type") == "checkbox") {
7               value = String($(element).attr("checked"));
8               param = param.toLowerCase();
9           }
1
0               return (value == param);
1       });
1
1       jQuery.validator.unobtrusive.adapters.addSingleVal("equal", "valuetocompa
  ;
```

Первые три строчки с комментариями необходимы только поддержки IntelliSense и указывают на используемые файлы с описаниями функций.

Затем, с помощью вызова *addMethod(),* в список проверок добавляется новое правило. Обратите внимание, что указывается тоже самое имя, которое было задано в атрибуте. Это обязательно, за исключением случаев, когда правило содержит не более одного параметра. Причина этого заключена в адаптерах и будет рассмотрена в следующей части.

Последним параметром *addMethod()* является функция, ответственная за проверку значения. При вызове она получает три параметра:

- *value* – текущее значение поля ввода;

- *element* – элемент страницы;
- *param* – список параметров, переданный атрибутом проверки данных.

Код созданной функции проверки сравнивает введенное значение с заданным в *param*. Данная реализация учитывает особенность объектов представляющих checkbox, которые содержат значение не в HTML атрибуте value, а в checked. Метод при успешной проверке возвращает *true*, в противном случае – *false*.

jQuery Validation обработает значение, полученное от функции проверки. Если оно будет равно *false*, то на месте блока , созданного методом *ValidationMessageFor()* и отмеченного при этом специальным HTML атрибутом, будет выведен текст сообщения об ошибке. Оно будет убрано в дальнейшем, после успешного прохождения проверки.

Остается разобрать еще одну, последнюю, строчку из созданного JavaScript кода.

Адаптер для jQuery Validate

Рассматривая код функции проверки может возникнуть вопрос: а откуда возьмётся значение для сравнения в переменой *param*? Логично предположить, что оно будет передано из jQuery Validation. Но откуда тогда сама библиотека возьмет это значение и текст сообщения об ошибке?

Для этой цели существуют адаптеры, которые считывают значения из атрибутов HTML тегов и передают их в *jQuery.validator*. Они располагаются в объекте *jQuery.validator.unobtrusive.adapters* и являются дополнением для jQuery Validation, созданным разработчиками ASP.NET MVC 3.

Посмотрим еще раз на последнюю строку созданного JavaScript:

```
?
1;
        jQuery.validator.unobtrusive.adapters.addSingleVal("equal", "valuetocompar
```

Метод *addSingleVal()* используется для адаптации правил с одним значением. Имя адаптера указано в качестве первого и совпадает с именем правила. Затем следует название параметра (ключ из заданной в C# коде пары), значение которого будет передано как переменная *param* в созданную выше функцию.

Разумеется, не все правила используют только одно значение. Существуют другие вариант добавления адаптеров, которые будет рассмотрены в следующей части.

Подключение JavaScript в Представление

Остался один небольшой шаг: необходимо подключить созданный JavaScript на страницу создания профиля. Это можно сделать вручную добавив соответствующую строку в Представление UserProfiles\Create. Или

же перетащить в него файл EqualAttribute.js из окна Solution Navigator (Solution Explorer). Для указания пути воспользуемся вызовом метода *Url.Content()*. Результат будет следующим:

```
1
2
3
        @model MVCDemo.Models.NewUserProfileModel
4
        @using MVCDemo.Resources.Views.UserProfiles;
5
        @{
6
           ViewBag.Title = CreateRes.PageTitle;
7
        }
8
9
        <h2>@ViewBag.Title</h2>
1
0
        <script src="@Url.Content("~/Scripts/jquery.validate.min.js")" type="text/j
1
        <script src="@Url.Content("~/Scripts/jquery.validate.unobtrusive.min.js")"
1
        <script src="@Url.Content("~/Scripts/Validation/EqualAttribute.js")" type=
1
2
        @using (Html.BeginForm()) {
1
        .....
3
1
4
```

Теперь можно запустить веб-приложение посмотреть на новую функциональность в работе. Приступим к разработке аналогичной поддержки проверки на стороне клиента и для оставшегося атрибута.

Проверка на стороне клиента для атрибута [StringLengthRange]

Реализация атрибута пойдет по знакомому уже плану, но в этот раз без создания адаптера.

Добавим классу *StringLengthRangeAttribute* поддержку интерфейса *IClientValidatable*. Здесь необходимо отметить, что jQuery Validation уже содержит набор часто используемых правил. Для их использования в ASP.NET MVC 3 существуют специальные версии классы, наследники *ModelClientValidationRule()*.

```
?
1       namespace MVCDemo.Attributes.Validation
2       {
3          using System;
4          using System.Collections.Generic;
5          using System.ComponentModel.DataAnnotations;
6          using System.Web.Mvc;
7
8          [AttributeUsage(AttributeTargets.Property, AllowMultiple = false, Inheri
9          public class StringLengthRangeAttribute : ValidationAttribute, IClientVa
1          {
```

```
        private readonly int _minLength;
        private readonly int _maxLength;

        .....

        #region IClientValidatable Members

        public IEnumerable<ModelClientValidationRule> GetClientValidation
            ModelMetadata metadata, ControllerContext context)
        {
            yield return new ModelClientValidationStringLengthRule(
                this.FormatErrorMessage(metadata.DisplayName),
                this._minLength,
                this._maxLength);
        }

        #endregion
    }
}
```

В данном случае нет необходимости в создании клиентской части на JavaScript и добавлении ссылки на неё в Представление. Использование *ModelClientValidationStringLengthRule()*позволяет использовать уже готовые адаптер и реализацию алгоритма проверки длины строки.

Стоит отметить, что использование готовых правил позволяет избежать их двойного применения к объекту. Дело в том, что атрибуты с одинаковыми стандартными правилами будут использовать их одинаковые имена. А это привет к ошибке в момент обращения к Представлению на этапе выполнения.

Классы ASP.NET MVC для стандартных правил

- *ModelClientValidationEqualToRule* – устанавливает равенство значений текущего и указанного поля.
- *ModelClientValidationRangeRule* – определяет минимальное и максимальное значения свойства.
- *ModelClientValidationRegexRule* – проверяет значение на соответствие регулярному выражению. Используется стандартным атрибутом *[RegularExpression]*.
- *ModelClientValidationRemoteRule* – используется для удаленной проверки данных. Используется стандартным ASP.MET MVC 3 атрибутом *[Remote]*.
- *ModelClientValidationRequiredRule* – указывает что данное поле обязательно к заполнению. Используется стандартным атрибутом *[Required]*.
- *ModelClientValidationStringLengthRule* – ограничивает длину заданной строки.

Осталась только одна проверка, выполняемая только на сервере. Это тест на уникальность выбранного имени входа на сайт. И опять для данного значения будет использован свой подход. Но перед тем как приступить к реализации, посмотрим на стандартные адаптеры ASP.NET MVC 3 для jQuery Validation.

Задание

Провести проверку на стороне любого кода JavaScript.

Практическая работа № 9 «Проверки на стороне сервера»

Цель: - научиться проводить проверки на стороне сервера

Форма отчета:
−выполнить задание;
−показать преподавателю;
−ответить на вопросы преподавателя.
Время выполнения: 2 ч

Есть клиент и есть сервер, общаются по именованому каналу (named pipe) Проблема: нужно сделать так чтобы сервер после записи данных в поток проверял прочитал ли их клиент и если не прочитал то давал ему на это время (допустим 500 мс, думаю этого достаточно) и если за это время клиент так и не считал данные то сервер разрывал подключение дабы поток не простаивал в ожидании. Проблема в том что просто запись в поток всегда ждёт пока клиент завершит чтение (хоть он там и пол часа висеть будет и не захочет читать с потока данные), а асинхронная запись происходит сразу же и не даёт никакого доп. времени клиенту на чтение. Из тех методов что нашёл у NamedPipeServerStream есть: pipeServer.WaitForPipeDrain(); - никакого толку, ждёт завершение всех операций, те по сути считай что асинхронная запись превратилась в синхронную и опять возможен висяк потока на долгое время WriteTimeout / ReadTimeout; - тоже никакого толку так как выбрасывается исключение что NamedPipeServerStream не поддерживает таких вещей. Ещё конечно же остался вариант в сервере после записи данных написать:

```
1.  if(pipeServer.IsConnected)
2.      Thread.Sleep(500);
```

Но это точно же не выход так как сервер будет точно висеть 500 мс, а клиент то может и через 10 мс уже давно всё считал с потока и его нужно было давным-давно закрыть. Как в таких случаях поступают подскажите пожалуйста, ато сам я решение найти не смог. Код моего сервера и клиента (асинхронные операции не используются пока что). Моя обёртка через которую происходит чтение и запись в поток при общении между клиентом и сервером:

Копировать

```
1.  //класс для работы с потоком
2.  //Позволяет писать пары логин-пароль и считывать результаты их обработки
3.  class StreamRW
4.  {
5.      private Stream ioStream;//поток куда писать данные
6.
7.      public StreamRW(Stream stream)
8.      { ioStream = stream; }
9.
10.     //запись пары логин-пароль в поток
11.     public void WriteData(Tuple<string, string> data)
12.     {
13.         byte[] outBuffer;
14.         using (MemoryStream m = new MemoryStream())
15.         using (BinaryWriter writer = new BinaryWriter(m))
16.         {
17.             writer.Write(data.Item1);
18.             writer.Write(data.Item2);
19.             outBuffer = m.ToArray();
20.         }
21.         int len = outBuffer.Length;
22.         ioStream.WriteByte((byte)(len / 256));
23.         ioStream.WriteByte((byte)(len & 255));
24.         ioStream.Write(outBuffer, 0, len);
25.         ioStream.Flush();
26.     }
27.
28.     //чтение пары логин-пароль из потока
29.     public Tuple<string, string> ReadData()
30.     {
31.         Tuple<string, string> result;
32.         int len;
33.         len = ioStream.ReadByte() * 256;
34.         len += ioStream.ReadByte();
35.         byte[] inBuffer = new byte[len];
36.         ioStream.Read(inBuffer, 0, len);
37.         using (MemoryStream m = new MemoryStream(inBuffer))
38.         using (BinaryReader reader = new BinaryReader(m))
39.         {
40.             try
41.             {
42.                 result = new Tuple<String, String>(reader.ReadString(), reader.ReadString());
43.             }
```

```
44.                 catch (IOException ex)
45.                 {
46.                     Console.WriteLine("Error has occured :(\n" +
47.                         "If you'd like to know more, you can google it:" +
ex.Message);
48.                     result = new Tuple<String, String>("Error", "Error");
49.                 }
50.             }
51.             return result;
52.         }
53.
54.         //запись кода результата обработки
55.         public void WriteInt(int answer)
56.         {
57.             byte[] outBuffer = new byte[1] { Convert.ToByte(answer) };
58.             ioStream.Write(outBuffer, 0, 1);
59.             ioStream.Flush();
60.         }
61.
62.           //чтение кода результата обработки (если соединение
разорвано на стороне сервера то вернёт -1).
63.         //Исключение выброшено не будет даже если поток закрыт!
64.         public int ReadInt()
65.         {
66.             return Convert.ToInt32(ioStream.ReadByte());
67.         }
68.     }
```

Сервер:
Копировать

```
1.         private static void ServerThread(object data)
2.             {
3.                 //создаём именованный канал
4.                 NamedPipeServerStream pipeServer =
5.                                     new NamedPipeServerStream("testpipe",
PipeDirection.InOut,          numThreads,          PipeTransmissionMode.Byte,
PipeOptions.Asynchronous);
6.
7.                 //пара логин-пароль для получения инфо от клиента
8.                 Tuple<string, string> logpass;
9.
10.                 //получим ID (просто для вывода на экран, больше не
используется)
11.                 int threadId = Thread.CurrentThread.ManagedThreadId;
12.
```

```
13.                 //сделаем вечный цикл чтобы сервер работал постоянно (ну почти постоянно)
14.             while (true)
15.             {
16.                 //подождём подключения
17.                 pipeServer.WaitForConnection();
18.                   //когда клиент подключился выведем уведомление об этом
19.                  Console.WriteLine("Client connected on thread[{0}].", threadId);
20.                 try
21.                 {
22.                     //настроим обёртку на канал
23.                     StreamRW srw = new StreamRW(pipeServer);
24.
25.                     //получим связку логин-пароль
26.                     logpass = srw.ReadData();
27.
28.                     //покажем их на экран
29.                      Console.WriteLine("Get from client on thread[{0}]: " + logpass.Item1 + " " + logpass.Item2, threadId);
30.
31.                     //отправим ответ клиенту
32.                     srw.WriteInt(1);
33.
34.                     //выведем инфо что клиент отключился
35.                             Console.WriteLine("Client on thread[{0}] disconnected.", threadId);
36.                 }
37.                 catch (IOException e)
38.                 {
39.                     Console.WriteLine("ERROR: {0}", e.Message);
40.                 }
41.                 //отсоединить клиента ибо нефиг тут висеть
42.                 pipeServer.Disconnect();
43.             }
44.             pipeServer.Close();
45.         }
```

Клиент:
Копировать

```
1.             {
2.                 NamedPipeClientStream pipeClient =
3.                     new NamedPipeClientStream(".", "testpipe",
4.                         PipeDirection.InOut, PipeOptions.None,
5.                         TokenImpersonationLevel.Impersonation);
```

```
6.
7.                  Console.WriteLine("Connecting to server...\n");
8.                  pipeClient.Connect();
9.
10.                 //настроить обёртку на канал
11.                 StreamRW srw = new StreamRW(pipeClient);
12.
13.                 //записать связку логин-пароль в поток
14.                                                 srw.WriteData(new
Tuple<string,string>("login","password"));
15.
16.                  //"зависнуть" на 5секунд чтобы продемонстрировать
что сервер всё это время тоже висит и ничего не делает
17.                       //в ожидании пока клиент не считает ответ  об
обработке данных (число int)
18.                 Thread.Sleep(5000);
19.
20.                 //вывести ответ от сервера
21.                 Console.Write(srw.ReadInt());
22.
23.                 pipeClient.Close();
24.                   // Give the client process some time to display results
before exiting.
25.                 Thread.Sleep(4000);
26.             }
```

Практическая работа № 10 «Настройки доступа к репозиторию»

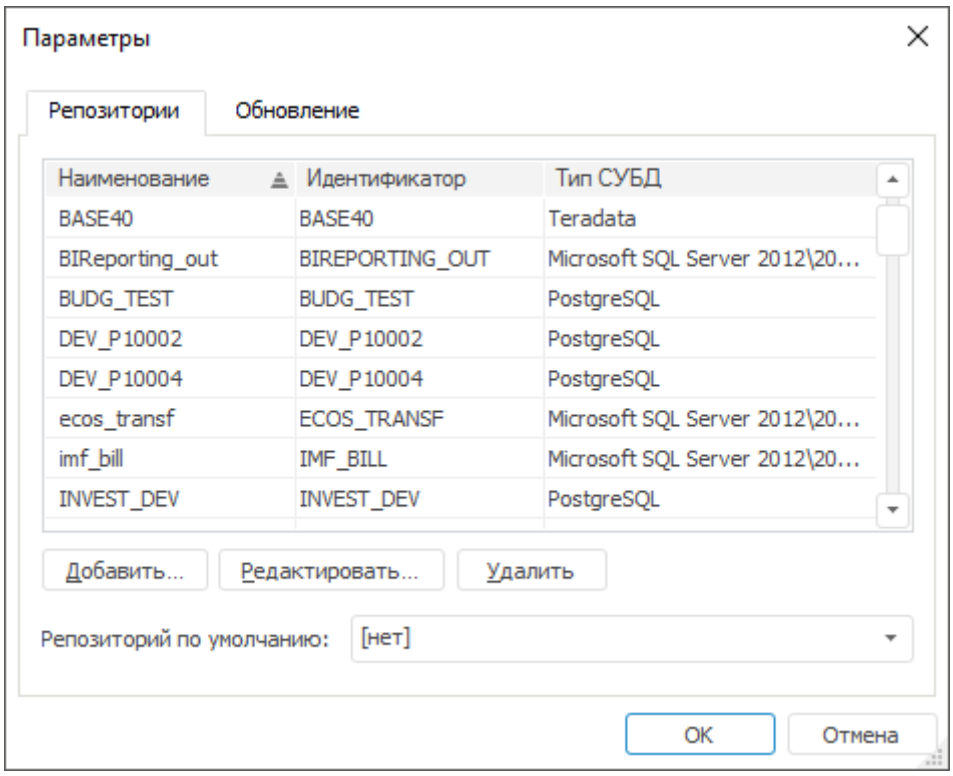
Цель: - научиться настраивать доступ к репозиторию

Форма отчета:
−выполнить задание;
−показать преподавателю;
−ответить на вопросы преподавателя.
Время выполнения: 2 ч

Работа со списком репозиториев в настольном приложении

Список репозиториев формируется на вкладке «Репозитории» диалога «Параметры».

- Для открытия диалога «Параметры»



- Для подключения репозитория с сервера СУБД нажмите кнопку «Добавить» и укажите параметры создаваемого репозитория
- Для редактирования параметров выбранного репозитория:
  - нажмите кнопку «Редактировать»;
  - выполните команду контекстного меню «Редактировать»;
  - дважды щёлкните по репозиторию в списке;
  - нажмите клавишу ENTER.
- Для удаления выбранных репозиториев из списка:
  - нажмите кнопку «Удалить»;
  - выполните команду контекстного меню «Удалить»;
  - нажмите клавишу DELETE.

Поле «Репозиторий по умолчанию» предназначено для выбора репозитория, который будет отображаться по умолчанию при запуске «Форсайт. Аналитическая платформа».

Задание

Установить доступ к репозиторию на своем ПК.

Практическая работа № 11 «Использование метрик программного продукта»

Цель: - научиться использовать метрик программного продукта

Форма отчета:
−выполнить задание;
−показать преподавателю;
−ответить на вопросы преподавателя.
Время выполнения: 2 ч

В настоящее время в программной инженерии еще не сформировалась окончательно система метрик. Действуют разные подходы к определению их набора и методов измерения. Система измерения включает метрики и модели измерений, которые используются для количественной оценки качества ПО.

При определении требований к ПО задаются соответствующие им внешние характеристики и их атрибуты (субхарактеристики), определяющие разные стороны управления продуктом в заданной среде. Для набора характеристик качества ПО, приведенных в требованиях, определяются соответствующие метрики, модели их оценки и диапазоны их значений для измерения отдельных атрибутов качества.

Согласно стандарту 1БО 14598 метрики определяются по модели измерения атрибутов ПО на всех этапах ЖЦ (промежуточная, внутренняя метрика), и особенно на этапе тестирования или функционирования (внешние метрики) продукта. Остановимся на классификации существующих метрик ПО, правилах проведения метрического анализа и процессах их измерения.

Существует три типа метрик: метрики программного продукта, которые используются при измерении его характеристик или свойств; метрики процесса, которые используются при измерении свойств процесса ЖЦ создания продукта; метрики использования.

Метрики программного продукта. Эти метрики используют внешние метрики, обозначающие свойства продукта, видимые пользователю, и внутренние метрики, обозначающие свойства, видимые только команде разработчиков.

Внешние метрики программного продукта:

- • метрики надежности, которые служат для определения числа дефектов;
- • метрики функциональности, с помощью которых устанавливаются наличие и правильность реализации функций в продукте;
- • метрики сопровождения, с помощью которых измеряются ресурсы продукта (скорость, память, среда);
- • метрики применимости продукта, которые способствуют определению степени доступности для изучения и использования;
- • метрики стоимости, которыми определяется стоимость созданного продукта.

Специальной мерой может служить уровень повторного использования компонентов программной системы, измеряемый как отношение размера продукта, изготовленного из готовых компонентов, к размеру системы в целом. Данная мера используется также при определении стоимости и

качества ПО. В качестве примеров таких метрик можно привести следующие характеристики: общее число объектов и число повторно используемых; общее число повторно используемых и новых операций; число классов, наследующих специфические операции; число классов, от которых зависит данный класс; число пользователей класса/операций и др.

При оценке общего количества некоторых величин часто используются среднестатистические метрики (среднее число операций в классе, наследников класса или операций класса и др.). Примером таких широко используемых внешних метрик являются метрики Холстеда — это характеристики программ, выявляемые на основе статической структуры программы на конкретном языке программирования, например число вхождений наиболее часто встречающихся операндов и операторов, длина программы как сумма числа вхождений всех операндов и операторов и др. На основе этих атрибутов можно вычислить время программирования, уровень программы (структурированность, качество) и языка программирования (уровень абстракции используемых средств языка, степень ориентации на проблему) и др.

Как правило, используемые метрики в значительной степени являются субъективными и зависят от знаний экспертов, производящих количественные оценки атрибутов компонентов программного продукта.

Метрики процесса. В качестве этих метрик могут быть использованы такие, как время разработки, число ошибок, найденных на этапе тестирования, и др. Но на практике обычно широко используются следующие метрики процесса:

- общее время разработки и отдельно время для каждой стадии;
- время модификации моделей;
- время выполнения работ на процессе;
- число найденных ошибок при инспектировании;
- стоимость проверки качества;
- стоимость процесса разработки.

Метрики использования. Они служат для измерения степени удовлетворения потребностей пользователя при решении его задач, помогают оценить не свойства самой программы, а результаты ее эксплуатации — ее эксплуатационное качество. Примерами могут служить точность и полнота реализации задач пользователя, затраченные ресурсы на эффективное решение задач пользователя (трудозатраты, производительность и др.).

Стандартная оценка показателей качества. В соответствии с рассмотренной четырехуровневой моделью качества оценка качества ПО начинается с нижнего уровня иерархии, т.е. с самого элементарного свойства оцениваемого атрибута показателя качества согласно установленным мерам. На этапе проектирования устанавливают значения оценочных элементов для каждого атрибута показателя качества анализируемого ПО, включенного в требования.

По определению стандарта 180/1E89126-2 метрика качества ПО представляет собой «модель измерения атрибута, связываемого с

показателем его качества». При измерении показателей качества ПО стандарт 180/1E89126-2 рекомендует использовать следующие типы мер:

• меры размера в разных единицах измерения (количество функций, размер программы, объем ресурсов и др.);

• меры времени, периоды реального, процессорного или календарного времени (время функционирования системы, время выполнения компонента, время использования и др.);

• меры усилий, продуктивное время, затраченное на реализацию проекта (производительность труда отдельных участников проекта, коллективная трудоемкость и др.);

• меры интервалов между событиями, например время между последовательными отказами;

• счетные меры, счетчики для определения количества обнаруженных ошибок, структурной сложности программы, числа несовместимых элементов, числа изменений (например, число обнаруженных отказов и др.).

Метрики качества используются при оценке качества программы (безотказной работы, выполнимости функций, удобства применения интерфейсов пользователей, БД и т.п.) с помощью данных, полученных после проведения испытаний на множестве тестов.

При тестировании наиболее важным показателем является наработка на отказ, который как атрибут надежности определяет среднее время между появлением угроз, нарушающих безопасность, и обеспечивает трудноизмеримую оценку ущерба, которая наносится соответствующими угрозами.

Очень часто оценка программы проводится по числу строк. При сопоставлении двух программ, реализующих одну и ту же прикладную задачу, предпочтение отдается более короткой, так как ее создает более квалифицированный персонал, в ней меньше скрытых ошибок, ее легче модифицировать и времени на отладку и модификацию уходит меньше, хотя по стоимости она, как правило, дороже. Таким образом, длину программы можно использовать для сравнения и оценки программ с учетом квалификации разработчиков, стиля разработки и используемой среды.

Если в требованиях к ПО было указано использовать несколько показателей, то каждый просчитанный после сбора данных показатель умножается на соответствующий весовой коэффициент, а затем все показатели суммируются для получения комплексной оценки уровня качества ПО. На основе измерения количественных характеристик и проведения экспертизы качественных показателей с применением весовых коэффициентов вычисляется итоговая оценка качества продукта путем суммирования результатов по отдельным показателям и сравнения их с эталонными показателями ПО (стоимость, время, ресурсы и др.).

При проведении оценки отдельного показателя с помощью оценочных элементов просчитываются весовой коэффициент-метрика, коэффициент-показатель, коэффициент-атрибут. Например, в качестве показателя возьмем переносимость. Этот показатель будет вычисляться по пяти известным атрибутам, причем каждый из них будет умножаться на соответствующий коэффициент. Все метрики-атрибуты суммируются и образуют показатель

качества. Когда все атрибуты оценены по каждому из показателей качества, производится суммарная оценка отдельного показателя, а потом и интегральная оценка качества с учетом весовых коэффициентов всех показателей ПО.

В конечном итоге результат оценки качества является критерием эффективности и целесообразности применения используемых методов проектирования, инструментальных средств и методик оценивания результатов создания программного продукта на стадиях ЖЦ.

Согласно стандарту ДСТУ 3230-1995 для оценки значений показателей качества используются следующие методы: измерительный, регистрационный, расчетный и экспертный (а также комбинации этих методов).

Измерительный метод основан на использовании измерительных и специальных программных средств для получения информации о характеристиках ПО, например определения объема, числа строк кода, операторов, количества ветвей в программе, числа точек входа/ выхода, реактивности и др.

Регистрационный метод используется при подсчете времени, числа сбоев или отказов, начала и конца работы ПО в процессе его выполнения.

Расчетный метод базируется на статистических данных, собранных при проведении испытаний, эксплуатации и сопровождении ПО. Расчетными методами оцениваются показатели надежности, точности, устойчивости, реактивности и др.

Экспертный метод осуществляется группой экспертов — специалистов, компетентных в решении данной задачи или используемом ПО. Их оценка базируется на опыте и интуиции, а не на результатах расчетов и экспериментов. Такая экспертиза обычно проводится путем просмотра программ и сопроводительных документов; для этого устанавливаются контролируемые признаки, которые коррелиро-ваны с одним или несколькими показателями качества и включены в опросные карты экспертов. Метод применяется при оценке таких показателей, как анализируемость, документируемость, структурированность ПО, и способствует всесторонней и качественной оценке созданного продукта.

При оценке значений показателей качества в зависимости от особенностей используемых ими свойств, способов их определения и назначения для каждой метрики качества применяется определенная шкала измерений:

• шкала метрическая (абсолютная, относительная, интегральная);

• шкала порядковая (ранговая), позволяющая ранжировать характеристики путем сравнения с опорными значениями;

• классификационная шкала, характеризующая наличие или отсутствие рассматриваемого свойства у оцениваемого ПО.

Показатели, которые вычисляются с помощью метрических шкал, называются количественными, а показатели, определяемые с помощью порядковых и классификационных шкал, — качественными.

Стандарт 180/1E89126-2 рекомендует к применению пять видов шкал измерения и порядок их использования от менее строгой оценки к более строгой.

1. Номинальная шкала отражает категории свойств оцениваемого объекта без их упорядочения.

2. Порядковая шкала служит для упорядочения характеристики по возрастанию или убыванию путем сравнения их с базовыми значениями.

3. Интервальная шкала задает существенные свойства объекта (например, календарная дата).

4. Относительная шкала задает некоторое значение относительно выбранной единицы.

5. Абсолютная шкала указывает на фактическое значение величины (например, число ошибок в программе равно 10).

Практическая работа № 12 «Проверка целостности программного кода»

Цель: - научиться проверять целостность программного кода

Форма отчета:

−выполнить задание;

−показать преподавателю;

−ответить на вопросы преподавателя.

Время выполнения: 2 ч

В практической работе необходимо определить контрольные данные с использованием следующих способов:

- битов четности. В качестве исходных данных принять битовое представление букв фамилии в соответствии с кодировкой Windows 1251;

Буква	Битовая строка	Паритетный бит	
		четный (even)	нечетный (odd)

- контрольных цифр. В качестве исходных данных принять необходимое количество цифр (за исключением контрольной) из строки, состоящей из кодов букв фамилии, имени и отчества согласно их положению в алфавите:

- по алгоритму Луна (15 цифр);

- для штрихкода по стандарту EAN-13 (12 цифр);

- для ИНН физического лица (10 цифр);

- для кодов станций на железнодорожном транспорте (5 цифр);

- контрольных сумм (CRC). В качестве исходных данных принять коды 1-ой, 2-ой и 3-ей буквы своей фамилии согласно их положению в алфавите для порождающего полинома - $G(x) = x^4 + x^1 + x^0$.

- кода коррекции ошибок (ECC). В качестве исходных данных принять первые 11 битов первых двух буквы своей фамилии в соответствии с кодировкой Windows 1251. Рассчитать вектора контрольных битов и синдромов, а также паритетные биты при отсутствии ошибки, одиночной и двойной ошибке.

При оформлении отчета необходимо привести необходимые таблицы, исходные данные, расчеты и результаты.

Практическая работа № 13 «Анализ потоков данных»

Цель: - научиться анализировать поток данных

Форма отчета:
−выполнить задание;
−показать преподавателю;
−ответить на вопросы преподавателя.
Время выполнения: 2 ч

Объект изучения: методология DFD. Исследование: освоение метода структурного анализа потоков данных (DFD) в нотации Gane-Sarson (синтаксис графического языка, семантика, правила построения графических диаграмм, словаря данных и спецификации процессов). Инструментарий: • Microsoft Visual Professional 2002/2003; • Computer Associates BPwin 4.0.

Первым этапом работы является ознакомление с системой NetCracker и получение начальных навыков работы с ней (добавление устройств и установление связей между ними при создании проекта, настройка протоколов, определение интенсивности трафика, анализ загруженности сети и т.п.); при этом удобно использовать имеющийся проект TUTOR.NET, специально предназначенный для целей обучения.

На втором этапе студент получает от преподавателя индивидуальное задание для создания проекта и анализа работы сети. Типовыми заданиями являются следующие:

Задание 1. Составить модель компьютерной сети определенного подразделения вуза (например, вычислительной лаборатории кафедры), проанализировать потоки данных, определить «слабые места» (работающие на пределе возможностей устройства и/или линии связи) сети.

Задание 2. Составить модель компьютерной сети по заданию преподавателя (исходные данные – число подразделений эксплуатирующей сеть организации, число ПЭВМ в каждой комнате и усредненный трафик каждой, расстояния между комнатами, желательность дальнейшего расширения и т.д.). Заданием является анализ потоков данных, выявление «слабых мест», уровня живучести сети (например, наличие «обходных путей» при выходе из строя отдельных связей при использовании маршрутизации).

В отчете указываются заданные преподавателем исходные данные для проектирования сети, описывается выбранная стратегия создания сети, параметры сетевых устройств и линий связи, приводится схема (желательна печатная копия рабочей зоны главного окна системы NetCracker) сети и один из стандартных видов выходной информации (обычно Bill Of Materials).

Контрольные вопросы

1. Какие задачи проектирования и исследования сетей могут быть решены с использованием пакета NetCracker?
2. Для каких целей служит браузер устройств? Рабочая зона? Панель изображений?
3. Что такое многоуровневый проект?
4. Какие средства NetCracker позволяют количественно судить о степени загруженности конкретного канала связи?
5. Каким образом в системе NetCracker можно решать задачи типа «а что, если…»?

Практическая работа № 14 «Использование метрик стилистики»

Цель: - научиться использовать метрики стилистики

Форма отчета:

−выполнить задание;

−показать преподавателю;

−ответить на вопросы преподавателя.

Время выполнения: 2 ч

Следующие пять характеристик являются продолжением метрики Холстеда.

1. Для измерения теоретической длины программы **N'** М. Холстед вводит аппроксимирующую формулу:

N'= n1*log2(n1) + n2*log2(n2)

где n1 - словарь операторов; n2 - словарь операндов программы.

Вводя эту оценку, Холстед исходит из основных концепций теории информации, по аналогии с которыми частота использования операторов и операндов в программе

пропорциональна двоичному логарифму количества их типов. Таким образом, выражение представляет собой идеализированную аппроксимацию **N**, т. е. справедливо для потенциально корректных программ, свободных от избыточности или несовершенств (стилистических ошибок). Несовершенствами можно считать следующие ситуации:

а) последующая операция уничтожает результаты предыдущей без их использования;

б) присутствуют тождественные выражения, решающие совершенно одинаковые задачи;

в) одной и той же переменной назначаются различные имена и т. п.

Подобные ситуации приводят к изменению N без изменения n.

М. Холстед утверждает, что для стилистически корректных программ отклонение в оценке теоретической длины **N'** от реальной **N** не превышает 10%.

Можно использовать **N'** как эталонное значение длины программы со словарем n. Длина корректно составленной программы N, т. е. программы, свободной от избыточности и имеющей словарь n, не должна отклоняться от теоретической длины программы **N'** более чем на 10%. Таким образом, измеряя **n1**, **n2**, **N1** и **N2** и сопоставляя значения **N** и **N'** для некоторой программы, при более чем 10%-ном отклонении можно говорить о наличии в программе стилистических ошибок, т. е. несовершенств.

На практике **N** и **N'** часто существенно различаются.

2. Другой характеристикой, принадлежащей к метрикам корректности программ, по М. Холстеду, является уровень качества программирования L (уровень программы):

L=V'/V

где V и V' /определены выше.

Исходным для введения этой характеристики является предположение о том, что при снижении стилистического качества программирования уменьшается содержательная нагрузка на каждый компонент программы и, как следствие, расширяется объем реализации исходного алгоритма. Учитывая это, можно оценить качество программирования на основании степени расширения текста относительно потенциального объема V'. Очевидно, для идеальной программы **L**=1, а для реальной - всегда **L**<1.

3. Нередко целесообразно определить уровень программы, не прибегая к оценке ее теоретического объема, поскольку список параметров программы часто зависит от реализации и может быть искусственно расширен. Это приводит к увеличению метрической характеристики качества программирования. М. Холстед предлагает аппроксимировать эту оценку выражением, включающим только фактические параметры, т. е. параметры реальной программы: **L'= (2 n2)/ (n1*N2)** - уровень качества программирования, основанный лишь на параметрах реальной программы без учета теоретических параметров,

4. Располагая характеристикой **L'**, Холстед вводит характеристику I, которую рассматривает как интеллектуальное содержание конкретного алгоритма, инвариантное по отношению к используемым языкам реализации: **I = L' * V**.

По мнению самого автора, термин интеллектуальность не совсем удачен. Преобразуя выражение можно получить:

I = L'V = LV = V'V/V = V'.

Эквивалентность **I** и **V'** свидетельствует о том, что мы имеем дело с характеристикой информативности программы.

Введение характеристики I позволяет определить умственные затраты на создание программы. Процесс создания программы условно можно представить как ряд операций:

1) осмысление предложения известного алгоритма;

2) запись предложения алгоритма в терминах используемого языка программирования, т. е. поиск в словаре языка соответствующей инструкции, ее смысловое наполнение и запись.

Используя эту формализацию в методике Холстеда, можно сказать, что написание программы по заранее известному алгоритму есть N'-кратная выборка операторов и операндов из словаря программы n, причем число сравнений (по аналогии с алгоритмами сортировки) составит **log2(n).**

Если учесть, что каждая выборка-сравнение содержит, в свою очередь, ряд мысленных элементарных решений, то можно поставить в соответствие содержательной нагрузке каждой конструкции программы сложность и число этих элементарных решений. Количественно это можно характеризовать с помощью характеристики **L**, поскольку **1/L** имеет смысл рассматривать как средний коэффициент сложности, влияющий на скорость выборки для данной программы. Тогда оценка необходимых интеллектуальных усилий по написанию программы может быть измерена как.

Задание на лабораторную работу:

1. Скачать калькулятор любого производителя или взять разработанный студентами.

2. Сравнить два программных продукта: калькулятор фирмы Microsoft и калькулятор, написанный студентами (скачанный). Сравнение проводить по следующим оценочным элементам: надежность ПС, сопровождаемость, корректность. Критерии оценки (1 или 0)

3. Все сравнение занести в следующую таблицу

Код элемента	Наименование	Метод оценки	Оценка калькулятора фирмы Microsoft	Оценка калькулятора
Оценочные элементы фактора «Надежность ПС»				
	Наличие требований к программе по устойчивости функционирования при наличии ошибок во входных данных	Экспертный		
	Возможность обработки ошибочных ситуаций			
	Полнота обработки ошибочных данных			
	Наличие тестов для проверки допустимых значений входных данных			
	Наличие системы контроля полноты входных данных			
	Наличие средств контроля корректности входных данных			
	Наличие требований к программе по восстановлению процесса выполнения в случае сбоя ОС, внешних устройств, процессора			
	Наличие требований к программе по восстановлению результатов при отказах ОС, внешних устройств, процессора			
	Наличие средств восстановления при сбоях оборудования			
	Наличие возможности повторного старта с точки прерывания			
	Наличие обработки неопределенностей			

Практическая работа № 15 «Выполнение измерений характеристик кода в среде VisualStudio»

Цель: - Приобрести навыки программирования основных действий СУБД в MS VisualStudio .NET

Форма отчета:

−выполнить задание;

−показать преподавателю;

−ответить на вопросы преподавателя.

Время выполнения: 2 ч

- **ОБЕСПЕЧЕННОСТЬ ЗАНЯТИЯ**

Оборудование лаборатории и рабочих мест лаборатории:
компьютеры, принтер, сканер, проектор, программное обеспечение общего и профессионального назначения, комплект учебно-методической документации.
Реализация профессионального модуля предполагает обязательную учебную практику.
Оборудование и технологическое оснащение рабочих мест:
1.Комплект ТС компьютера IBM-PC
2.Методические указания для выполнения практических работ

4. Microsoft Visio.
5. Microsoft Visual Studio.
6.Microsoft Office.

- **КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ.**

Проверка, входит ли узел в поддерево, определяемое своим корнем (например, входит ли данный товар в группу одного из верхних уровней, "Кисточка" в "Инструменты для ремонта"):

```
WITH Поддерево ([Код территории], [Код вышестоящей территории], Наименование, Уровень)
AS (
SELECT [Код территории], [Код вышестоящей территории], Наименование, 1
FROM Территории
WHERE [Код территории] = 40288000 -- узел, проверяемый на вхождение

UNION ALL

SELECT Территории.[Код территории], Территории.[Код вышестоящей территории], Территории.Наименование, Уровень + 1
FROM Территории
INNER JOIN Поддерево ON Территории.[Код территории] = Поддерево.[Код вышестоящей территории]
)
SELECT result = CASE
WHEN EXISTS (
SELECT 1
FROM Поддерево
WHERE [Код территории] = 40260000 /* корень поддерева */
)
THEN 'Узел входит в поддерево'
ELSE 'Узел НЕ входит в поддерево'
END
```

Выборка поддерева по заданному узлу:

```
SELECT [Код подмножества], Уровень
FROM Подмножества
WHERE [Код множества] = 123 -- корень поддерева
ORDER BY Уровень
```

Выборка всех предков (путь к узлу от корня):

```
SELECT [Код множества], Уровень
FROM Подмножества
WHERE [Код подмножества] = 345 -- узел
ORDER BY Уровень
```

Проверка, входит ли узел в поддерево:

```
SELECT result = CASE
WHEN EXISTS (
SELECT 1
FROM Подмножества
WHERE [Код подмножества] = 345 /* узел */
AND [Код множества] = 211 /* корень поддерева */
)
THEN 'Узел входит в поддерево'
ELSE 'Узел НЕ входит в поддерево'
END
```

- **ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ РАБОТЫ**

1. Изучите методические указания и конспект лекций.
2. Проанализируйте задание по своему варианту.
3. Создайте приложение, отвечающее запросу задания.

- **МЕТОДИКА АНАЛИЗА РЕЗУЛЬТАТОВ, ОБРАЗЕЦ ОТЧЕТА.**

1. Отчет должен содержать цель работы.
2. Содержание индивидуального задания.
3. Перечень элементов, выбранных для создания приложения.

Список литературы

1. *Лаврищева, Е. М.* Программная инженерия и технологии программирования сложных систем : учебник для вузов / Е. М. Лаврищева. — 2-е изд., испр. и доп. — Москва : Издательство Юрайт, 2019. — 432 с. — (Бакалавр. Академический курс). — ISBN 978-5-534-07604-2. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://biblio-online.ru/bcode/436514
2. Информационные системы в экономике : учебник для академического бакалавриата / В. Н. Волкова, В. Н. Юрьев, С. В. Широкова, А. В. Логинова ; под редакцией В. Н. Волковой, В. Н. Юрьева. — Москва : Издательство Юрайт, 2019. — 402 с. — (Бакалавр и специалист). — ISBN 978-5-9916-1358-3. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://biblio-online.ru/bcode/436469
3. *Богатырев, В. А.* Информационные системы и технологии. Теория надежности : учебное пособие для бакалавриата и магистратуры / В. А. Богатырев. — Москва : Издательство Юрайт, 2019. — 318 с. — (Бакалавр и магистр. Модуль). — ISBN 978-5-534-00475-5. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://biblio-online.ru/bcode/433723
4. *Дибров, М. В.* Компьютерные сети и телекоммуникации. Маршрутизация в ip-сетях в 2 ч. Часть 1 : учебник и практикум для среднего профессионального образования / М. В. Дибров. — Москва : Издательство Юрайт, 2019. — 333 с. — (Профессиональное образование). — ISBN 978-5-534-04638-0. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://biblio-online.ru/bcode/437357
5. *Нестеров, С. А.* Информационная безопасность : учебник и практикум для академического бакалавриата / С. А. Нестеров. — Москва : Издательство Юрайт, 2019. — 321 с. — (Университеты России). — ISBN 978-5-534-00258-4. — Текст : электронный // ЭБС Юрайт [сайт]. — URL: https://biblio-online.ru/bcode/434171