Государственное бюджетное профессиональное образовательное
учреждение Республики Хакасия
Техникум коммунального хозяйства и сервиса

Фонд оценочных средств
профессионального модуля
**ПМ.02 Технческая поддержка и администрирование информационных ресурсов**
программы подготовки квалифицированных рабочих, должности служащих
по специальности
09.02.09 Веб-разработка

## I. Паспорт фонда оценочных средств

### 1.1 Общие положения

Результатом освоения профессионального модуля является готовность обучающегося к выполнению вида профессиональной деятельности по ПМ.02 Технческая поддержка и администрирование информационных ресурсов, и составляющих его профессиональных компетенций, а также общие компетенции, формирующиеся в процессе освоения ОПОП в целом.

В рамках оценочных материалов результатов освоения рабочей программы осуществляется оценка результатов практической подготовки обучающихся.

Оценка результатов практической подготовки осуществляется в образовательной организации (в техникуме) и (или) на предприятии, в организации.

Формой аттестации по профессиональному модулю является экзамен по модулю.

### 1.2 Формы промежуточной аттестации по профессиональному модулю

| **Элементы модуля, профессиональный модуль** | **Формы промежуточной аттестации** |
|---|---|
| 1 | 2 |
| МДК.02.01 Настройка и сопровождение информационных ресурсов | Комплексный экзамен |
| МДК.02.02 Обеспечение безопасности информационных ресурсов | |
| УП.01 Учебная практика | Дифференцированный зачёт |
| ПП.01 Производственная практика | Дифференцированный зачёт |
| **ПМ.02 Техническая поддержка и администрирование информационных ресурсов** | **Экзамен по модулю** |

## 2. Область применения фонда оценочных средств

Фонд оценочных средств предназначен для оценки результатов освоения
МДК.02.01 Настройка и сопровождение информационных ресурсов
МДК.02.02 Обеспечение безопасности информационных ресурсовТаблица 1

| **Наименование объектов контроля и оценки** (объекты оценивания) | **Основные показатели оценки результата и их критерии** | **Тип задания; № задания** | **Форма аттестации (в соответствии с учебным планом)** |
|---|---|---|---|
| *1* | *2* | *3* | *4* |
| **Профессиональные компетенции** | | | |
| ПК 2.1. Устанавливать прикладное программное обеспечение и модулей информационных ресурсов, включая их настройку. | **Владеть навыками**<br>- подготовки программной среды для функционирования веб-приложения;<br>**Уметь**<br>- соблюдать процедуру установки прикладного программного обеспечения в соответствии с документацией;<br>- идентифицировать инциденты, возникающие при установке программного обеспечения, и принимать решение по изменению процедуры установки;<br>- пользоваться нормативно-технической документацией в области программного обеспечения;<br>- производить настройку параметров веб-сервера;<br>- устанавливать систему управления базами данных (СУБД);<br>**Знать**<br>- принципы устройства и функционирования информационных ресурсов;<br>- принципы устройства и функционирования программных средств и платформ для разработки веб-ресурсов;<br>- инструменты и методы коммуникаций;<br>- каналы коммуникаций;<br>- модели коммуникаций;<br>- технологии межличностной и групповой коммуникации в | практическое задание №2, 4, 5,6 тестирование задание №1 самостоятельная работа задание № 3; задания к экзамену по МДК №7; Задание экзамена по модулю № 8 | Экзамен по модулю |

| | | | |
|---|---|---|---|
| | деловом взаимодействии, основ конфликтологии. | | |
| ПК 2.2. Проводить работы по резервному копированию и развертыванию резервной копии информационных ресурсов. | **Владеть навыками**<br>- организации и обеспечения функционирования подсистемы резервного копирования и восстановления;<br>**Уметь**<br>- выполнять регламентные процедуры по резервированию данных;<br>- устанавливать прикладное программное обеспечение для резервирования информационных ресурсов;<br>**Знать**<br>- современные стандарты взаимодействия компонентов распределенных приложений;<br>- возможности ИР; | практическое задание №2, 4, 5,6<br>тестирование задание №1<br>самостоятельная работа задание № 3;<br>задания к экзамену по МДК №7;<br>Задание экзамена по модулю № 8 | |
| ПК 2.3. Настраивать права пользователей в соответствии с функциональными задачами (ролями) и на основании информации о поведенческих факторах. | **Владеть навыками**<br>- настройки прав доступа пользователя в существующей системе;<br>**Уметь**<br>- идентифицировать права пользователей в зависимости от функционала информационного ресурса;<br>- регламентировать уровни прав и ролей пользователей информационных ресурсов;<br>- применять регламентные процедуры управления правами доступа пользователей информационных ресурсов;<br>**Знать**<br>- принципы использования электронно-цифровых подписей и работы удостоверяющих центров | практическое задание №2, 4, 5,6<br>тестирование задание №1<br>самостоятельная работа задание № 3;<br>задания к экзамену по МДК №7;<br>Задание экзамена по модулю № 8 | Экзамен по модулю |
| ПК 2.4. Применять программные средства обеспечения безопасности информации веб приложений. | **Владеть навыками**<br>- работы с инструментами мониторинга безопасности ИР;<br>- выполнения типовых регламентных процедур по защите ИР;<br>**Уметь**<br>- пользоваться нормативно- | практическое задание №2, 4, 5,6<br>тестирование задание №1<br>самостоятельная работа задание № 3;<br>задания к экзамену по МДК №7; | Экзамен по модулю |

| | | | |
|---|---|---|---|
| | технической документацией в области программного обеспечения;<br>**Знать**<br>- основы информационной безопасности веб-ресурсов; | Задание экзамена по модулю № 8 | |
| **Общие компетенции** | | | |
| ОК1. Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам. | распознавать задачу и/или проблему в профессиональном и/или социальном контексте; анализировать задачу и/или проблему и выделять её составные части; определять этапы решения задачи; выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы;<br>составить план действия; определить необходимые ресурсы;<br>владеть актуальными методами работы в профессиональной и смежных сферах; реализовать составленный план; оценивать результат и последствия своих действий (самостоятельно или с помощью наставника) | практическое задание<br>тестирование | Экзамен по модулю |
| ОК2. Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности. | определять задачи для поиска информации; определять необходимые источники информации; планировать процесс поиска; структурировать получаемую информацию; выделять наиболее значимое в перечне информации; оценивать практическую значимость результатов поиска; оформлять результаты поиска | практическое задание<br>тестирование | Экзамен по модулю |
| ОК3. Планировать и реализовывать собственное профессиональное и личностное развитие. | определять актуальность нормативно-правовой документации в профессиональной деятельности; применять современную научную профессиональную терминологию; определять и выстраивать траектории профессионального развития и самообразования | практическое задание<br>тестирование | Экзамен по модулю |
| ОК4. Работать в коллективе и команде, эффективно | организовывать работу коллектива и команды; взаимодействовать с коллегами, руководством, клиентами в ходе | практическое задание<br>тестирование | Экзамен по модулю |

| | | | |
|---|---|---|---|
| взаимодействовать с коллегами, руководством, клиентами. | профессиональной деятельности | | |
| ОК5.Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста. | грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке, проявлять толерантность в рабочем коллективе | практическое задание тестирование | Экзамен по модулю |
| ОК 06. Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе традиционных общечеловеческих ценностей, применять стандарты антикоррупционного поведения. | описывать значимость своей профессии (специальности) | практическое задание тестирование | Экзамен по модулю |
| ОК7. Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях. | соблюдать нормы экологической безопасности; определять направления ресурсосбережения в рамках профессиональной деятельности по профессии (специальности) | практическое задание тестирование | Экзамен по модулю |
| ОК8. Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержание необходимого уровня физической подготовленности. | использовать физкультурно-оздоровительную деятельность для укрепления здоровья, достижения жизненных и профессиональных целей; применять рациональные приемы двигательных функций в профессиональной деятельности; пользоваться средствами профилактики перенапряжения характерными для данной профессии (специальности) | практическое задание тестирование | Экзамен по модулю |
| ОК9. Использовать информационные технологии в | применять средства информационных технологий для решения профессиональных задач; | практическое задание тестирование | Экзамен по модулю |

| | | | |
|---|---|---|---|
| профессиональной деятельности. | использовать современное программное обеспечение | | |
| ОК10. Пользоваться профессиональной документацией на государственном и иностранном языке | участвовать в диалогах на знакомые общие и профессиональные темы; строить простые высказывания о себе и о своей профессиональной деятельности; кратко обосновывать и объяснить свои действия (текущие и планируемые); писать простые связные сообщения на знакомые или интересующие профессиональные темы | практическое задание тестирование | Экзамен по модулю |
| ОК 11. Использовать знания по финансовой грамотности, планировать предпринимательскую деятельность в профессиональной сфере. | использовать знания по финансовой грамотности, планировать предпринимательскую деятельность в профессиональной сфере. | практическое задание тестирование | Экзамен по модулю |

**3. Фонд оценочных средств**

## ЗАДАНИЕ 1.
## ЗАДАНИЯ ДЛЯ ТЕКУЩЕГО КОНТРОЛЯ

**ЗАДАНИЕ 1. МДК.01.01. Настройка и сопровождение информационных ресурсов**
**Тема 1.1. Установка прикладного программного обеспечения и модулей информационных ресурсов, включая их настройку**

### ЗАДАНИЕ 1.1. (теоретическое)
**Тема 1.1. Установка прикладного программного обеспечения и модулей информационных ресурсов, включая их настройку**

Тест 1

**Условия выполнения задания**

1. Место (время) выполнения задания: Учебный кабинет
2. Максимальное время выполнения задания: 25 мин.

***Инструкция:***

1.На выполнение теста отводиться 25 минут
2.Внимательно прочитайте вопрос
3.Выберите (впишите) правильный ответ
4.Один правильный ответ равен одному баллу

***Тест по основам инntернет***

*Выберите один вариант ответа:*

**Вопрос №1**

*Понятие "телекоммуникация" означает ...*

1) проверку работоспособности компьютера
**2) обмен информацией на расстоянии**
3) одно из важнейших свойств модема

**Вопрос №2**

*Протоколы компьютерных сетей - это ...*

1) сетевые программы, которые ведут диалог между пользователем и компьютером
**2) стандарты, определяющие формы представления и способы передачи сообщений**
3) различные марки компьютеров

**Вопрос №3**

*Одна из важнейших характеристик модема является ...*

**1) скорость передачи данных**
2) длина сетевого кабеля
3) вид передаваемой информации

**Вопрос №4**

*Для подключения компьютера в уже существующую локальную сеть необходимо, как минимум, следующий набор средств:*

1) модем, телефон и кабель
2) звуковая карта и автоответчик
**3) сетевая карта, кабель**

**Вопрос №5**

*Центральный компьютер, предоставляющий остальным компьютерам локальной*

*сети сервисы и данные, называется ...*

1) рабочей станцией

2) последовательным портом связи

**3) сервером**

**Вопрос №6**

*Совокупность условий и правил обмена информацией называется ...*

1) выделенным каналом связи

2) компьютерной сетью

**3) протоколом**

**Вопрос №7**

*Компьютерные сети, действующие в пределах одного какого-либо помещения, предприятия, учреждения, называют ...*

**1) локальными**

2) региональными

3) глобальными

**Вопрос №8**

*Выберите верное высказывание:*

1) принципы функционирования всех компьютерных сетей совершенно одинаковы

**2) для компьютерных коммуникаций используются коммутируемые телефонные линии**

3) максимальную скорость передачи обеспечивают все существующие модемы

**Вопрос №9**

*Современные модемы не обеспечивают ...*

1) прием и передачу факсимильных сообщений

2) автоматическое соединение с модемом на другом конце линии

**3) анализ полученной информации и вычисления с ее использованием**

**Вопрос №10**

*Задача любой компьютерной сети заключается в ...*

1) согласовании работы всех компонентов каждого компьютера

2) получении и отправки корреспонденции

**3) обмене информацией между компьютерами**

**Вопрос №11**

*Для передачи информации в локальных сетях обычно используют ...*

1) телефонную сеть

2) спутниковую связь

**3) кабель "витая пара"**

**Вопрос №12**

*Выберите верное высказывание:*

**1) к кабелю передачи данных подключено каждое устройство сети**

2) локальные компьютерные сети не ограничивают расстояние между соединенными компьютерами

3) кабель передачи данных не обязательно должен быть подключен к сетевой карте

**Вопрос №13**

*Одна из важнейших характеристик компьютерной сети является ...*

1) стоимость сетевого оборудования

2) вид передаваемой информации

**3) скорость передачи данных**

**Вопрос №14**

*Выберите неверное высказывание:*

1) рабочей станцией называется любой компьютер

2) сервер обслуживает всех пользователей сети

**3) в компьютерных сетях могут использоваться только одинаковые компьютеры**

**Вопрос №15**

*Совокупность условий и правил обмена информацией называется ...*

1) выделенным каналом связи

2) компьютерной сетью

**3) протоколом**

**Вопрос №16**

*Электронная поста позволяет передавать ...*

1) только почтовые сообщения

2) видеоизображения

**3) почтовые сообщения и приложенные к ним файлы**

**Вопрос №17**

*Глобальные компьютерные сети дают возможность ...*

1) организовать совместное использование ресурсов, а также общение множества пользователей, расположенных сравнительно недалеко друг от друга

**2) организовать обмен данными на больших расстояниях**

3) передавать электроэнергию на очень большие расстояния

**Вопрос №18**

*Сетевые серверы - это ...*

**1) узлы связи на базе мощных компьютеров, обеспечивающие круглосуточную передачу информации**

2) стандартные декодирующие устройства, с помощью которых любой компьютер может подключиться к глобальной сети

3) различные персональные компьютеры, связанные с разными организациями

**Вопрос №19**

*Выберите верное высказывание:*

1) по электронной почте можно вести только частную переписку

2) с помощью Интернета невозможно получить доступ к файлам на компьютерах, расположенных в других странах

**3) с глобальной сетью тесно связаны понятия киберпространства и виртуальной реальности**

**Вопрос №20**

*Гипертекст - это ...*

**1) структурированный текст, в котором могут осуществляться переходы по выделенным ссылкам**

2) текст, введенный с клавиатуры в память компьютера

3) текст, в котором используется очень сложный шифр

**Вопрос №21**

*Организация, предоставляющая услуги по подключению к Интернету*

*пользовательских персональных компьютеров, называется ...*

1) браузером
**2) провайдером**
3) рабочей станцией

**Вопрос №22**

*Глобальная компьютерная сеть не позволяет ...*

1) передавать изображения в реальном времени
**2) обеспечивать электропитанием рабочую станцию или сервер**
3) передавать различные речевые сообщения

**Вопрос №23**

*Выберите верное высказывание:*

**1) первая компьютерная сеть была создана в США в 1969 г.**
2) глобальная сеть является одноранговой
3) модем производит вычисления согласно

**Вопрос №24**

*Имеется адрес электронной почты в сети Интернет: user newname@int.glasnet.ru. Каково имя владельца этого электронного адреса?*

1) int.glasnet.ru
2) user_newname
**3) glasnet.ru**

**Вопрос №25**

*Узлы связи на базе мощных компьютеров, обеспечивающих круглосуточную передачу информации, - это...*

1) стандартные декодирующие устройства
**2) сетевые серверы**
3) любые персональные компьютеры

**Вопрос №26**

*Поисковые системы общего назначения позволяют находить документы в WWW ...*

**1) по ключевым словам**
2) по назначениям протоколов
3) по ASCII - кодам

**Вопрос №27**

*Организация, которым необходимо предоставить широкий доступ к своим хранилищам файлов, могут сделать это, используя ...*

1) WWW
**2) FTP**
3) электронную почту

**Вопрос №28**

*Укажите сервис, устанавливающий расстояние, ради которого десятки миллионов людей становятся пользователями Интернета:*

1) HTTP - сервер
2) FTP - сервер
**3) e-mail**

**Вопрос №29**

*Для отправления почтового сообщения по электронной почте надо обязательно*

*указать ...*

1) файловые вложения
2) текст письма
**3) адрес почтового ящика**

**Вопрос №30**

*Выберите неверное высказывание:*

1) программное обеспечение для работы с Интернетом развивается очень быстро
**2) отличие гипертекста состоит в том, что формат его хранения и передачи не является стандартным для всей сети**
3) доступ к магазинам электронной торговли обычно организован с помощью гипертекстовых страниц

**Ответы:**

1) 2;
2) 2;
3) 2;
4) 2;
5) 2;
6) 2;
7) 2;
8) 2;
9) 2;
10) 1;
11) 1;
12) 2;
13) 1;
14) 1;
15) 1;
16) 1;
17) 1;
18) 1;
19) 2;
20) 1.

Критерии оценки теста:
«5» - 86-100% правильных ответов на вопросы (26 и более правильных ответов)
«4» - 71-85% правильных ответов на вопросы (21 – 25 правильных ответов)
«3» - 51-70% правильных ответов на вопросы (15-20 правильных ответов)
«2» - 0-50% правильных ответов на вопросы (менее 15 правильных ответов)

**ЗАДАНИЕ 1.2 (теоретическое)**

Тема 1.2. Основы проектирования сайтов
Тест 1

**Условия выполнения задания**

1. Место (время) выполнения задания: Учебный кабинет
2. Максимальное время выполнения задания: 25 мин.

***Инструкция:***

1.На выполнение теста отводиться 25 минут

2.Внимательно прочитайте вопрос
3.Выберите (впишите) правильный ответ
4.Один правильный ответ равен одному баллу

**Тест: "** Основы проектирования сайтов **".**

Тестируемый: _____ Дата: _____

| **Задание №1** | | |
|---|---|---|
| **Документы, содержащие гиперссылки и предназначенные для просмотра в браузере, называются ...** | | |
| Выберите один из 4 вариантов ответа: | | |
| 1) | | веб-сервер |
| 2) | | веб-страницы |
| 3) | | веб-клиент |
| 4) | | веб-сайт |

| **Задание №2** | | |
|---|---|---|
| **На веб-страницу можно поместить ...** | | |
| Выберите несколько из 6 вариантов ответа: | | |
| 1) | | Тексты |
| 2) | | Рисунки |
| 3) | | Видеоизображения |
| 4) | | Файлы |
| 5) | | Гиперссылки |
| 6) | | Папки |

| **Задание №3** | | | |
|---|---|---|---|
| **Установите соответствие:** | | | |
| | | | |
| 1) | Файл\Создать | А) | Оформление шрифта страницы |
| 2) | Файл\Сохранить | Б) | Оформление фона с помощью рисунка |
| 3) | Вставка\Рисунок | В) | Вставка таблицы |
| 4) | Таблица\Вставить | Г) | Вставка рисунка |
| 5) | Формат\Шрифт | Д) | Создание веб-страницы |
| 6) | Формат\Фон | Е) | Сохранение веб-страницы |

| **Задание №4** | | |
|---|---|---|
| **Выберите стандартный тип файла веб-страницы** | | |
| Выберите несколько из 6 вариантов ответа: | | |
| 1) | | doc |
| 2) | | txt |

| | | |
|---|---|---|
| 3) | | html |
| 4) | | jpg |
| 5) | | exe |
| 6) | | htm |

| Задание №5 | | |
|---|---|---|
| **Тематически связанные веб-страницы обычно бывают представлены форме ...** | | |
| Выберите один из 4 вариантов ответа: | | |
| 1) | | веб-клиента |
| 2) | | веб-сайта |
| 3) | | веб-документа |
| 4) | | веб-сервера |

| Задание №6 | | |
|---|---|---|
| **Какие программы являются браузерами?** | | |
| Выберите несколько из 5 вариантов ответа: | | |
| 1) | | Internet Explorer |
| 2) | | WinRAR |
| 3) | | Mozilla Firefox |
| 4) | | Microsoft Office PowerPoint |
| 5) | | Opera |

| Задание №7 | | |
|---|---|---|
| **Гипертекст - это ...** | | |
| Выберите один из 4 вариантов ответа: | | |
| 1) | | очень большой текст |
| 2) | | текст, набранный на компьютере |
| 3) | | текст, в котором используется шрифт большого размера |
| 4) | | структурированный текст, в котором могут осуществляться переходы по гиперссылкам |

| Задание №8 | | |
|---|---|---|
| **Гиперссылки на веб-странице могут обеспечить** | | |
| Выберите несколько из 4 вариантов ответа: | | |
| 1) | | переход только в пределах данной веб-страницы |
| 2) | | переход только на веб-страницы данного сервера |
| 3) | | открытие любого файла по указанному адресу |
| 4) | | переход на любую web-страницу любого сервера Интернет |

| Задание №9 | |
|---|---|
| **Расположите по порядку части URL-адреса веб-страницы** | |
| Укажите порядок следования всех 3 вариантов ответа: | |
| 1) | путь и имя файла |
| 2) | протокол  передачи гипертекста |
| 3) | имя сервера Интернета |

| Задание №10 | | |
|---|---|---|
| **Прикладной протокол, обеспечивающий передачу и отображение веб-страниц, - это:** | | |
| Выберите один из 4 вариантов ответа: | | |
| 1) | | HTTP |
| 2) | | FTP |
| 3) | | IP |
| 4) | | TCP |

| Задание №11 | | |
|---|---|---|
| Адрес файла, который представляет собой выражение вида: **протокол://доменный адрес сервера/имя файла,** называется | | |
| Выберите один из 4 вариантов ответа: | | |
| 1) | | URL |
| 2) | | WWW |
| 3) | | протокол |
| 4) | | IP-адрес |

| Задание №12 | | |
|---|---|---|
| **Что такое редактор визуального веб-конструирования?** | | |
| Выберите один из 4 вариантов ответа: | | |
| 1) | | Это система программирования которая позволяет решать численные задачи |
| 2) | | Язык гипертекстовой разметки |
| 3) | | язык спрограммирования на котором мы можем писаль различные программы |
| 4) | | Программа для создания веб-сайтов |

| Задание №13 | | |
|---|---|---|
| **Как можно осуществить просмотр веб-страницы, работая редактор визуального веб-конструирования?** | | |
| Выберите несколько из 5 вариантов ответа: | | |
| 1) | | Использовать **Конструктор** |
| 2) | | Использовать режим **С разделением** |
| 3) | | В режиме **Код** |

| | | |
|---|---|---|
| 4) | | С помощью режима **Просмотр** |
| 5) | | Используя команду **Файл\Просмотреть в обозревателе** |

**Тест: "** Основы проектирования сайтов **".**

**Ответы:**

| | |
|---|---|
| #1 (1 б.) | 2 |
| #2 (1 б.) | 1, 2, 3, 5 |
| #3 (1 б.) | 1=1, 2=2, 3=3, 4=4, 5=5, 6=6 |
| #4 (1 б.) | 3, 6 |
| #5 (1 б.) | 2 |
| #6 (1 б.) | 1, 3, 5 |
| #7 (1 б.) | 4 |
| #8 (1 б.) | 3, 4 |
| #9 (1 б.) | 1=1, 2=2, 3=3 |
| #10 (1 б.) | 1 |
| #11 (1 б.) | 1 |
| #12 (1 б.) | 4 |
| #13 (2 б.) | 4, 5 |

**Раздел 2. Разработка интерфейсов пользователя**
**МДК.01.02. Разработка интерфейсов пользователя**

**ЗАДАНИЕ 1.3 (теоретическое)**
**Тема 2.1. Разработки прототипов пользовательских интерфейсов**
Тестовое задание

**Условия выполнения задания**
1. Место (время) выполнения задания: Учебный кабинет
2. Максимальное время выполнения задания: 25 мин.

***Инструкция:***
1.На выполнение теста отводиться 25 минут
2.Внимательно прочитайте вопрос
3.Выберите (впишите) правильный ответ
4.Один правильный ответ равен одному баллу

**Тест по теме «Пользовательский интерфейс»**

**1. Формирование индивидуального информационного пространства:**
- a. установка программного обеспечения на персональный компьютер;
- b. создание текстовых, графических и других документов;
- c. перенос (копирование) на свой компьютер фотографий, фильмов, текстов, музыки;
- d. сохранение на своем компьютере ссылок на сетевые ресурсы;
- e. все выше перечисленное;

**2. Информационное пространство пользователя**
- a) пространство при переносе, копировании и сохранения ссылок;
- b) при решении задач по физике с помощью компьютера;
- c) это информационные ресурсы (файлы с программами, документами, веб-сайты, фотографии, видеофрагменты и др.), которые доступны пользователю при работе на ПК;
- d) при решении задач по геометрии с помощью создания документов;

e) рисование объектов;

**3. Совокупность средств и правил взаимодействия компьютера и человека:**

a) аппаратный интерфейс;
b) системный интерфейс;
c) человеческий рабочий интерфейс;
d) пользовательский интерфейс;
e) прикладной интерфейс;

**4. Можно выделить следующие типы окон:**

a) окна папок;
b) диалоговые окна;
c) окна приложений;
d) окна документов;
e) все выше перечисленное;

**5. Диалоговые окна предназначены для:**

a) для одностороннего взаимодействия человека и компьютера;
b) для диалога человека и компьютера;
c) для одностороннего взаимодействия компьютера и человека
d) для диалога человека и человека;
e) для диалога компьютера и компьютера;

**6. Основными элементами графического интерфейса являются:**

a) окна и меню;
b) папки и файлы;
c) рабочий стол и кнопка «Пуск»
d) программы;
e) окна приложений;

**7. Совокупность средств и правил взаимодействия пользователя с компьютером называют:**

a) аппаратным интерфейсом;
b) программным интерфейсом;
c) процессом;
d) объектом управления;
e) пользовательским интерфейсом

**8. Какие из перечисленных функций отображены кнопками состояния окна?**

a) свернуть, копировать, восстановить, закрыть;
b) свернуть, копировать, вставить;
c) вырезать, вставить, закрыть, копировать;
d) свернуть, развернуть, восстановить, закрыть;
e) вырезать, удалить, копировать, вставить;

**9. Объекты объектно-ориентированного графического интерфейса представляются в виде:**

a) иконок с картинками;
b) значков с рисунками;
c) иконок и значков;
d) заранее заданными частями экрана;
e) картинок с рисунками;

**10. Программы, с помощью которых пользователь решает свои информационные задачи, не прибегая к программированию, называются:**

a) драйверами;
b) сервисными программами;
c) прикладными программами;
d) текстовыми редакторами;
e) операционной системой;

11. **Комплекс программ, обеспечивающих совместное функционирование всех устройств компьютера и предоставляющих пользователю доступ к его ресурсам, — это:**

a) файловая система;
b) прикладные программы;
c) операционная система;
d) сервисные программы;
e) текстовыми редакторами;

**12. Основное окно операционной системы:**

a) окна и меню;
b) рабочий стол;
c) панель задач;
d) кнопка «Пуск»
e) программы и приложения;

**13. Совокупность всех программ, предназначенных для выполнения на компьютере, называют:**

a) системой программирования;
b) программным обеспечением;
c) операционной системой;
d) приложениями;
e) программами;

**14. Взаимодействие человека и компьютера строится на основе:**

a) объектного графического интерфейса;
b) ориентированного интерфейса;
c) объектно-ориентированного графического интерфейса;
d) простого интерфейса;
e) сложного интерфейса;

КЛЮЧ:

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| e | c | d | e | b | a | e | d | c | c | c | b | b | c |

## Раздел 3. Тестирование информационных ресурсов и интеграция программного кода

## МДК.01.03. Тестирование информационных ресурсов

### ЗАДАНИЕ 1.4 (теоретическое)

**Тема 3.1. Тестирование готового программного кода**

**Условия выполнения задания**

1. Место (время) выполнения задания: Учебный кабинет
2. Максимальное время выполнения задания: 25 мин.

***Инструкция:***

1.На выполнение теста отводиться 25 минут
2.Внимательно прочитайте вопрос
3.Выберите (впишите) правильный ответ
4.Один правильный ответ равен одному баллу

Тест "Методы тестирования и отладки. Документация программного обеспечения"

Вопрос 1

Каким стандартом определяется процесс создания документации пользователя всех видов для ПС, имеющего интерфейс пользователя

Варианты ответов

ГОСТ Р ИСО/МЭК 15910

ГОСТ ИСО/МЭК 15910

ГОСТ ИСО/МЭК 9126

ГОСТ Р ИСО/МЭК 9126

Вопрос 2

Документ, в котором формулируют основные цели разработки , требования к программному продукту, определяют сроки и этапы разработки и регламентируют процесс приема-сдаточных испытаний

Ответ записать полностью

Вопрос 3

В этом разделе Технического задания разделе указывают цель разрабатываемого программного продукта, краткую характеристику области применения программного обеспечения и объекта, в котором используют программное обеспечение.

Варианты ответов

Введение

Основания для разработки

Назначение разработки.

Требования к программе или программному изделию

Требования к программной документации

Технико-экономические показатели

Стадии и этапы разработки

Порядок контроля и приемки

Вопрос 4

В этом разделе Технического задания должно быть указано функциональное и эксплуатационное назначение программного обеспечения.

Варианты ответов

Введение.

Основания для разработки .

Назначение разработки.

Требования к программе или программному изделию.

Требования к программной документации.

Технико-экономические показатели.

Стадии и этапы разработки.

Порядок контроля и приемки.

Вопрос 5

В этом разделе Технического задания должны быть указаны:

- документ (документы), на основании которых ведется разработка;

- организация, утвердившая разработанный документ, и дата его утверждения;

- наименование и (или) условное обозначение темы разработки

программы.

Варианты ответов
Введение.
Основания для разработки .
Назначение разработки.
Требования к программе или программному изделию.
Требования к программной документации.
Технико-экономические показатели.
Стадии и этапы разработки.
Порядок контроля и приемки.
Вопрос 6
В этом разделе Технического задания должен быть приведен предварительный состав программной документации и , при необходимости, специальные требования к ней.

Варианты ответов
Введение.
Основания для разработки .
Назначение разработки.
Требования к программе или программному изделию.
Требования к программной документации.
Технико-экономические показатели.
Стадии и этапы разработки.
Порядок контроля и приемки.
Вопрос 7
Сколько подразделов имеет раздел Технического задания "Требования к программе или программному продукту"?

Варианты ответов
7
8
6
5
Вопрос 8
Согласно стандарту ГОСТ Р ИСО/МЭК 9126 качество программного обеспечения может быть оценено следующими характеристиками:

Варианты ответов
Функциональные возможности
Надежность
Эффективность
Безопасность
Современность
Сопровождаемость
Вопрос 9
Способность программного обеспечения сохранять свой уровень качества функционирования при установленных условиях за установленныи период времени.

Вопрос 10
Способность программного обеспечения быть перенесенным из одного окружения в другое

Вопрос 11

Процесс оценивания качества программного обеспечения состоит из трех стадии:

Расставьте стадии в правильном порядке

Варианты ответов

установление (определение) требований к качеству

подготовка к оцениванию

процедура оценивания

Вопрос 12

Определение оператора/операторов программы, выполнение которого вызвало нарушение вычислительного процесса.

Варианты ответов

Локализация

Отладка

Тестирование

Вопрос 13

Процесс локализации и исправления ошибок, обнаруженных при тестировании программного обеспечения.

Варианты ответов

Отладка

Локализация

Тестирование

Вопрос 14

Методы отладки программ:

Варианты ответов

Метод ручного тестирования

Метод индукции

Метод дедукции

Метод обратного прослеживания

Метод "Черного ящика"

Метод "Белого ящика"

Метод прямого прослеживания

Вопрос 15

Метод тестирования при котором тестировщик вводит данные и анализирует результат, но он не знает, как именно работает программа.

Варианты ответов

Метод индукции

Метод дедукции

Метод "Черного ящика"

Метод "Белого ящика"

Метод "Серого ящика"

Вопрос 16

Метод тестирования при котором тестировщик разрабатывает тесты , основываясь на знании исходного кода, к которому он имеет полный доступ.

Варианты ответов

Метод дедукции

Метод индукции
Метод "Черного ящика"
Метод "Белого ящика"
Вопрос 17
Это тестирование представляет собой сбор показателей времени отклика программного обеспечения на внешний запрос в целях определения производительности и установления соответствия требованиям, предъявляемым
к данной системе.

Варианты ответов
нагрузочное тестирование
стресс-тестирование
тестирование стабильности
конфигурационное тестирование
Вопрос 18
Это тестирование программного обеспечения , которое оценивает надежность и устоичивость системы в условиях превышения пределов нормального функционирования

Варианты ответов
нагрузочное тестирование
стресс-тестирование
тестирование стабильности
конфигурационное тестирование
Вопрос 19
Проверка работоспособности программного обеспечения при длительном тестировании с ожидаемым уровнем нагрузки.

Варианты ответов
нагрузочное тестирование
стресс-тестирование
тестирование стабильности
конфигурационное тестирование
Ответы

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 вариант | В | Б | А | В | Б | Б | В | А | Б | А |

Критерии оценки теста:
«5» - 86-100% правильных ответов на вопросы (26 и более правильных ответов)
«4» - 71-85% правильных ответов на вопросы (21 – 25 правильных ответов)
«3» - 51-70% правильных ответов на вопросы (15-20 правильных ответов)
«2» - 0-50% правильных ответов на вопросы (менее 15 правильных ответов)

## ЗАДАНИЕ 1.5 (теоретическое)

**Тема 4.1. Работа с системой контроля версий.**

**Условия выполнения задания**
1. Место (время) выполнения задания: Учебный кабинет
2. Максимальное время выполнения задания: 25 мин.

***Инструкция:***
1.На выполнение теста отводиться 25 минут

2.Внимательно прочитайте вопрос
3.Выберите (впишите) правильный ответ
4.Один правильный ответ равен одному баллу

**https://clck.ru/36CMkh**

**Инструкция по прохождению теста:**

Вам предоставлена одна попытка для прохождения теста.

Время попытки ограничено – до 10 минут.

Тест состоит из 10 заданий.

Внимательно читайте задание, затем отвечайте на вопрос.

После выполнения заданий теста Вы увидите кнопку:

***Посмотреть баллы*** – после нажатия этой кнопки Вы увидите правильные ответы, на какие вопросы даны неверные ответы и общее количество набранных баллов.

После завершения выполнения теста вы сможете просмотреть оценку.

| | Оценка 5 | Оценка 4 | Оценка 3 | Оценка 2 |
|---|---|---|---|---|
| Баллы | 10 | 8-9 | 6-7 | 5-0 |

Критерии оценки теста:
«5» - 86-100% правильных ответов на вопросы (26 и более правильных ответов)
«4» - 71-85% правильных ответов на вопросы (21 – 25 правильных ответов)
«3» - 51-70% правильных ответов на вопросы (15-20 правильных ответов)
«2» - 0-50% правильных ответов на вопросы (менее 15 правильных ответов)

## ЗАДАНИЕ 2.4 (теоретическое)

**Тема 1.3. Тестирование методом белого ящика.**
Задание
Условия выполнения задания
1. Место (время) выполнения задания: Учебный кабинет
2. Максимальное время выполнения задания: 25 мин.
Инструкция:

1.На выполнение теста отводиться 25 минут
2.Внимательно прочитайте вопрос
3.Выберите (впишите) правильный ответ
4.Один правильный ответ равен одному баллу

В ходе выполнения задания на тему «Тестирование условий» студент должен сначала написать консольное приложение (в соответствии с вариантом) на любом языке программирования (Pascal, Delphi, C++, C#, Java и т. п.) согласно постановке задачи. Далее необходимо провести тестирование консольного приложения способом тестирования ветвей и операций отношений согласно следующим шагам [1, 5, 6]:

1) Строится ограничение для каждого условия.
2) Выявляются ограничения результата по каждому простому условию.
3) Строится ограничивающее множество для каждого условия (с применением константных формул).
4) Для каждого элемента ограничивающих множеств разрабатывается тестовый вариант.
5) Реальные результаты каждого тестового варианта сравниваются с ожидаемыми результатами. Примеры задач:

1) Пользователь вводит два числа, и . Необходимо вычислить и вывести на экран значение по следующему принципу: а) Если одновременно и , тогда вычисляется как . б) Иначе, если , то вычисляется как произведение и . в) Иначе вычисляется как .
2) Пользователь вводит два числа, и . Необходимо вычислить и вывести на экран значение по следующему принципу: а) Если одновременно и , тогда вычисляется как среднее геометрическое модулей и . б) Иначе, если , то вычисляется как сумма и . в) Иначе вычисляется как . В ходе выполнения лабораторной работы на тему «Тестирование циклов» требуется написать для каждой из двух задач консольное приложение (в соответствии с вариантом) на любом языке программирования (Pascal, Delphi, C++, C#, Java и т. п.) согласно постановке задачи. Далее необходимо провести тестирование каждого консольного приложения способом тестирования циклов [1]:

1) Проанализировать конструкцию циклов и определиться с методикой тестирования.
2) Разработать тестовые варианты.
3) Реальные результаты каждого тестового варианта необходимо сравнить с ожидаемыми результатами. Массив в каждой задаче должен быть организован как псевдодинамический, т. е. реализуемый следующим образом:

1) объявляется массив, состоящий из 5 элементов;
2) пользователь вводит реальное количество элементов массива;
3) дальнейшая работа с массивом ограничивается заданной пользователем размерностью (элементы массива генерируются случайным образом). Примеры задач:

1) Вычислить и вывести на экран сумму и число положительных элементов прямоугольной матрицы, у которых индекс строки больше или равен индексу столбца.
2) Упорядочить по возрастанию элементы каждой строки прямоугольной матрицы. Измененную матрицу вывести на экран.
3) Дана прямоугольная матрица. Определить и вывести на экран количество элементов данной матрицы, которые больше суммы остальных элементов в своем столбце.
4) Найти наибольший и наименьший элементы прямоугольной матрицы и поменять их местами. Измененную матрицу вывести на экран.
5) Дана прямоугольная матрица. Найти в каждой строке матрицы максимальный и минимальный элементы и поменять их с первым и последним элементом строки соответственно. Измененную матрицу вывести на экран. Решение студентом задач, аналогичных рассмотренным выше, помогает освоить алгоритмы тестирования программного обеспечения, основанные на принципе «белого ящика».

## ЗАДАНИЕ №2
## ПРАКТИЧЕСКИЕ РАБОТЫ

**Раздел 1. Проектирование информационных ресурсов**
**МДК.01.01. Проектирование информационных ресурсов**
**Тема 1.1. Проектирование информационных ресурсов**

**Практическая работа № 6. Описание бизнес процессов заданной предметной области**

**Цель:** научиться описывать бизнес-процессы той или иной предметной области.

***Бизнес-процесс*** – это логическая последовательность действий человека (или нескольких человек) в коллективе. Цель описания бизнес-процесса – анализ и регламентация тех или иных действий в коллективе.

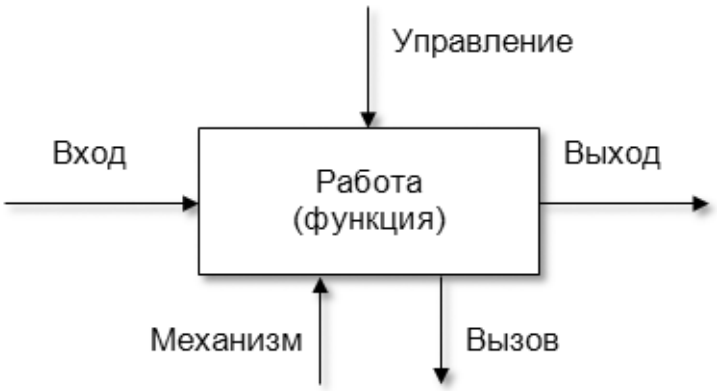



Рисунок 1 – элементы графической нотации IDEF0

***Предметная область*** – это часть реального мира, которая подлежит изучению с целью автоматизации организации управления.

Предметной областью информационной системы является совокупность объектов, свойства которых и отношения между которыми представляют интерес для пользователей ИС.

Любая предметная область может быть разбита на фрагменты. Каждый фрагмент оперирует со своими объектами и с множеством пользователей, которые имеют свои взгляды на предметную область, поэтому выявление

предметной области и ее анализ является неотъемлемой частью разработки любой информационной системы.

***Задание 1.*** Провести анализ предметной области

| Вариант | Предметная область | Краткое описание |
|---|---|---|
| **1** | **Страховая медицинская компания** | заключает договоры добровольного медицинского страхования с населением и договоры с лечебными учреждениями на лечение застрахованных клиентов. При возникновении страхового случая клиент подает заявку на оказание медицинских услуг по условиям договора инспектору, который работает с данным клиентом. Инспектор направляет данного клиента в лечебное учреждение. Отчеты о своей деятельности инспектор предоставляет в бухгалтерию. Бухгалтерия проверяет оплату договоров, перечисляет денежные средства за оказанные услуги лечебным учреждениям, производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. СМК не только оплачивает лечение застрахованного лица при возникновении с ним страхового случая, но и, при возникновении каких-либо осложнений после лечения, оплачивает лечение этих осложнений. |
| **2** | **Агентство недвижимости** | занимается покупкой, продажей, сдачей в аренду объектов недвижимости по договорам с их собственниками. Агентство управляет объектами недвижимости как физических, так и юридических лиц. Собственник может иметь несколько объектов. В случае покупки или аренды клиент может произвести осмотр объекта. Одной из услуг, предлагаемых агентством, является проведение инспектирования текущего состояния объекта для адекватного определения его рыночной цены. По результатам своей деятельности агентство производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **3** | **Фотоцентр** | занимается оказанием фотоуслуг и продажей различных фототоваров. В состав фотоуслуг входят: печать фотографий, проявление фотопленок, художественное |

| | | |
|---|---|---|
| | | фото, фото на документы, реставрация фотографий, выезд фотографа для съемки объекта. Поставка необходимых материалов осуществляется через дилеров ведущих мировых производителей фототоваров. Согласно отдельному договору, различные химические отходы передаются предприятию по утилизации вредных веществ. По результатам своей деятельности фотоцентр производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **4** | **Ателье** | занимается изготовлением одежды. Клиент может выбрать любую модель изделия из каталога, либо осуществить индивидуальный заказ. Отдельно с клиентом оговариваются материал, его свойства (цвет, прочность и т. д.), срочность выполнения заказа, даты примерок. После согласования всех деталей рассчитывается ориентировочная стоимость заказа, на основании которой клиент вносит аванс. После выполнения заказа клиент оплачивает его окончательную стоимость. По результатам своей деятельности ателье производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **5** | **Компания по разработке программных продуктов** | заключает договор с клиентом на разработку программного продукта согласно техническому заданию. После утверждения технического задания определяется состав и объем работ, составляется предварительная смета. На каждый проект назначается ответственный за его выполнение — куратор проекта, который распределяет нагрузку между программистами и следит за выполнением технического задания. Когда программный продукт готов, то его внедряют, производят обучение клиента и осуществляют дальнейшее сопровождение. По результатам своей деятельности компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **6** | **Кадровое агентство** | способствует трудоустройству безработных граждан. Агентство ведет учет и классификацию данных о безработных на основании резюме от них. От предприятий города поступают данные о свободных вакансиях, на основании которых агентство предлагает различные варианты |

| | | |
|---|---|---|
| | | трудоустройства соискателям. В случае положительного исхода поиска вакансия считается заполненной, а безработный становится трудоустроенным. По результатам своей деятельности кадровое агентство производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **7** | **Строительная организация** | занимается строительством объектов по заказам клиентов. Сначала заказ проходит предварительную стадию: сбор различных разрешений на строительство, составление эскиза объекта, расчет объема и закупка строительных материалов. Сами строительные материалы доставляются на объект партиями. По мере поступления очередной партии стройматериалов закладывается фундамент объекта, строится каркас здания. По результатам данной работы происходит согласование с заказчиком, после чего утепляется контур, вставляются окна, устанавливается крыша. Дальше идет обсуждение с клиентом внутренней отделки здания, закупаются отделочные материалы. После того, как объект проходит технический контроль, он передается заказчику. В дополнительные услуги строительной организации входят: услуги дизайнера по интерьеру, закупка и доставка мебели, сотрудничество с охранным предприятием по установке сигнализации. По результатам своей деятельности строительная организация производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **8** | **Ресторан** | предоставляет для своих клиентов услугу питания. На каждый день составляется меню, которое включает в себя список блюд для питания. На основе этого меню составляется список для закупки необходимых продуктов питания, входящих в состав блюд. Клиент, приехав в ресторан, выбирает из меню блюда, которые он хотел бы заказать, их готовят, если они заранее не были готовы, и приносят клиенту. В качестве дополнительной услуги ресторан может организовать развлекательные программы в своем помещении. По результатам своей деятельности ресторан производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |

| | | |
|---|---|---|
| **9** | **Отдел вневедомственной охраны (ОВО)** | занимается охраной объектов физических и юридических лиц. ОВО является коммерческим подразделением милиции. Клиент, желающий обеспечить охрану своего имущества, обращается в ОВО и составляет договор охраны. В договоре оговариваются следующие моменты: адрес объекта; план расположения помещений; количество входов/выходов; расположение окон; список лиц, отвечающих за имущество; ответственное лицо от клиента, которое будет присутствовать в момент вскрытия помещения. После заключения договора объект подключается к сигнализации. В случае срабатывания сигнализации дежурный посылает патруль на осмотр объекта и сообщает ответственному лицу клиента о данном факте. Патруль, вместе с ответственным лицом клиента, осматривает объект, проверяет сохранность имущества и работу сигнализации (в случае ложного срабатывания). После каждого выезда составляется акт, который является основанием для возбуждения уголовного дела относительно лиц, незаконно проникшим на объект. По результатам своей деятельности ОВО предоставляет отчетность в вышестоящие органы милицейского руководства. |
| **10** | **Обувная фабрика** | производит разнообразную обувь, ассортимент которой зависит от конъюнктуры рынка, от сезона, от моды. У различных поставщиков фабрика закупает необходимые для производства материалы и сырье. Готовая продукция отпускается в магазины под реализацию. При необходимости, магазины могут высказывать свои пожелания/претензии на ассортимент. Брак и отходы производства передаются специальному предприятию по утилизации. По результатам своей деятельности обувная фабрика производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **11** | **Мебельный центр** | занимается изготовлением мебели на заказ. Дизайнер приезжает к клиенту, замеряет необходимые параметры будущей мебели и составляет предварительную смету. Клиент вносит предоплату для закупки необходимых материалов. После изготовления мебели рассчитывается окончательная стоимость заказа, |

| | | |
|---|---|---|
| | | осуществляются доставка и сборка, происходит полный расчет за заказ. По результатам своей деятельности мебельный центр производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **12** | **Завод по производству напитков** | занимается производством и оптовой реализацией различных напитков. Клиент делает заказ на доставку партий напитков. В связи с тем, что производство является довольно длительным технологическим процессом (20–30 дней), заказы принимаются предварительно за месяц. В отделе менеджмента собираются все заказы на текущий месяц, рассчитывается необходимое количество сырья и материалов, составляется план работы производственного цеха. Готовые напитки поступают в отдел розлива, где упаковываются в тару и передаются на склад. По мере поступления готовой продукции на склад, рабочие склада развозят напитки заказчикам. По результатам своей деятельности завод по производству напитков производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **13** | **Компьютерная компания** | занимается продажей, ремонтом, сборкой, тестированием компьютерной техники. Также специалисты компании предоставляют услуги по разработке и монтажу локальных вычислительных сетей. Вся техника и комплектующие закупаются оптом у дилеров и хранятся на складе. Клиент, который хочет приобрести товар, оформляет заказ в торговом зале, а забирает технику со склада или оставляет заявку на ее доставку. Клиент, который хочет отремонтировать технику, приносит ее в сервисный отдел, откуда, по прошествии некоторого времени, забирает как отремонтированную или как технику, не подлежащую ремонту. По желанию клиента, специалисты компании могут выехать к клиенту для общей диагностики возникшей проблемы с техникой. По результатам своей деятельности компьютерная компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **14** | **Компания по предоставлению телекоммуникационных** | занимается оказанием телекоммуникационных услуг абонентам. Клиент делает заявку на подключение к |

| | | |
|---|---|---|
| | **услуг** | телекоммуникационным услугам и ему, по необходимости, устанавливают соответствующее оборудование. Оплата за услуги вносится путем авансовых платежей. Каждый факт предоставления услуги фиксируется соответствующим оборудованием и является основанием для списания соответствующей суммы с личного счета абонента. Клиент в любое время суток может получить отчет об оказанных ему услугах, их стоимости и остатку на личном счете абонента. По результатам своей деятельности компания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **15** | **Управляющая компания ЖКХ** | занимается обслуживанием жилого фонда города. УК получает финансовые средства от населения и бюджета города в виде компенсаций и субсидий на коммунальные услуги. На основании поступивших средств УК осуществляет текущий ремонт жилого фонда, а также капитальный ремонт согласно плану. Для непосредственного выполнения работ УК нанимает соответствующую рабочую силу (сантехников, дворников, электриков и т. д.). По результатам своей деятельности УК ЖКХ производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **16** | **Авиакомпания** | совершает авиаперелеты между городами. В зависимости от парка самолетов, сезона, спроса составляется расписание полетов. Данные о клиентах, купивших билеты на рейс, поступают из кассы. В случае неблагоприятных погодных условий рейс может быть отложен или отменен, о чем необходимо сообщить клиентам, которые могут отказаться от рейса или вылететь другим. В авиакомпании существует система скидок для постоянных клиентов, детей, своих сотрудников. По результатам своей деятельности авиакомпания производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **17** | **Хлебопекарня** | занимается производством хлеба и хлебобулочных изделий, которые выпекаются в специальном оборудовании — печи. Готовый хлеб развозится по различным торговым точкам города, с которыми у |

| | | |
|---|---|---|
| | | хлебопекарни заключен долгосрочный договор на поставку хлебобулочных изделий. Также любое физическое или юридическое лицо может сделать предварительный заказ на выпечку большой партии изделий на некоторое мероприятие. Хлебопекарня, в зависимости от объема хлебобулочных изделий для торговых точек и наличия предварительных заказов, закупает у поставщиков соответствующий объем сырья и материалов, а также составляет график работы персонала. По результатам своей деятельности хлебопекарня производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **18** | **Туроператор** | предоставляет возможность своим клиентам осуществить туристическую или деловую поездку в различные города России и мира. При разработке нового тура сначала анализируется текущая ситуация на рынке туризма и выбирается направление тура. После этого определяется статус тура, бронируются места в гостиницах и билеты на переезд к месту тура, разрабатывается культурная/деловая/развлекательная программа, утверждаются сроки тура. На каждый тур назначается ответственное лицо от туроператора, которое будет вести данный тур для улаживания проблем в случае возникновения каких-нибудь чрезвычайных или форс-мажорных ситуаций. Клиент приходит в офис туроператора, где вместе с менеджером выбирает уже разработанный тур и оформляет путевку. После возвращения из тура клиент может высказать свои замечания или пожелания, которые будут учтены при доработке существующих туров или при разработке новых. Также, для дальнейшего улучшения тура, туроператор проводит анализ отчетов от посредников (гостиница, гиды и т. д.). По результатам своей деятельности туроператор производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **19** | **Студия звукозаписи** | занимается поиском исполнителей песен различных жанров для записи, выпуска и продажи их альбомов. Продюсер исполнителя договаривается со студией о создании альбома. После подписания договора исполнитель записывает альбом. |

| | | |
|---|---|---|
| | | Когда альбом полностью записан, он отправляется в тираж. Копии альбома распределяются по торговым точкам. По результатам своей деятельности студия звукозаписи производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **20** | **Культурный центр** | занимается организацией и проведением различных массовых мероприятий (показ кино, театрализованные представления, различные шоу). В фойе здания проводятся различные выставки картин, музейных экспонатов. Каждое мероприятие разрабатывается самим центром или заказывается клиентом. На основе данных заказов формируется афиша на следующий месяц, составляются сценарии мероприятий, подбираются актеры. К конкретным мероприятиям, по возможности, заказываются определенные выставки, которые могут проходить и отдельно. По результатам своей деятельности культурный центр производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |
| **21** | **Спортивный комплекс** | предоставляет услуги по проведению спортивных тренировок. Тренировки, относящиеся к одному виду спорта, объединяются в спортивные секции. Клиент обращается в спортивный комплекс, где получает абонемент на посещение спортивной секции. На основе купленных абонементов составляется расписание тренировок на следующий месяц. Также, в зависимости от загруженности спортивного комплекса, распределяются тренеры спортивных секций. По результатам своей деятельности спортивный комплекс производит отчисления в налоговые органы и предоставляет отчетность в органы государственной статистики. |

НАПОМИНАНИЕ: анализ предметной области включает в себя организационную структуру предприятия (организации).

***Задание 2.*** Описать бизнес-процессы заданной предметной области.

***Задание 3.*** Оформить отчет.

НАПОМИНАНИЕ: отчет оформляется по ГОСТу, вывод писать обязательно (2-3 развернутых предложения).

**Практическое занятие № 2. «Построение графической нотаций на основе системного анализа и бизнес требований заказчика»**

Цель работы: освоить использование методологии IDEF0 для создания моделей бизнес-процессов.

## *Теоретическая справка*

Методологию IDEF0 можно считать следующим этапом развития графического языка описания функциональных систем *SADT* (Structured Analysis and Design Teqnique). IDEF0 стандарт был разработан в 1981 году в рамках программы автоматизации промышленных предприятий, которая носила обозначение *ICAM* (Integrated Computer Aided Manufacturing) и была предложена департаментом ВВС США. Семейство стандартов IDEF унаследовало свое обозначение от названия этой программы (IDEF=ICAM DEFinition). Отличием IDEF0 от SADT является усовершенствованный набор функций для описания бизнес-процессов и наличие эффективной методологии взаимодействия «аналитик-специалист», что позволило использовать новый метод для обеспечения групповой работы над созданием модели, с непосредственным участием аналитиков и специалистов, занятых в проекте.

*IDEF0* – графический язык моделирования бизнес-процессов, при помощи которого создаётся графико-текстовое описание системы, позволяющее дать ответ на некоторые заранее определённые вопросы, сама система при этом представляется в виде совокупности взаимодействующих работ или функций. Поэтому модели IDEF0 называются *функциональными моделями*. Такой подход позволяет анализировать функции системы безотносительно к объектам, для которых они применяются, благодаря чему гораздо проще смоделировать логику и взаимодействие бизнес-процессов.

В основе методологии IDEF0 лежат следующие правила и понятия:

– *функциональный блок* (*Activity Box*) – графически изображается в виде прямоугольника (см. рисунок 1) и представляет некоторую конкретную функцию или задачу в рамках рассматриваемой системы. По требованиям стандарта название каждого функционального блока должно быть сформулировано в глагольном наклонении (например, «*производить услуги*», или «*производство услуг*», но не «*произведённые услуги*»);

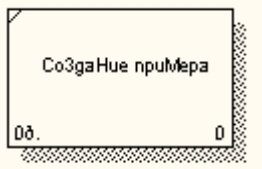



*Рисунок 1:* Функциональный блок

– *интерфейсные дуги* (*Arrays*) входят в функциональные блоки. Графически обозначаются в виде однонаправленной стрелки (смотрите рисунок 2). В зависимости от того, с какой стороны входит дуга, определяется её значение (роль); дуги, входящие в верхнюю сторону имеют значение «*Управление*» (*Control* в терминах методики); входящие слева имеют значение «*Вход*» (*Input*); входящие справа имеют значение «*Выход*» (*Output*); входящие снизу имеют значение «*Механизм»* (*Mechanism*); интерфейсные дуги также имеют название, которое обычно описывается существительным (например, *«Заявка»*).

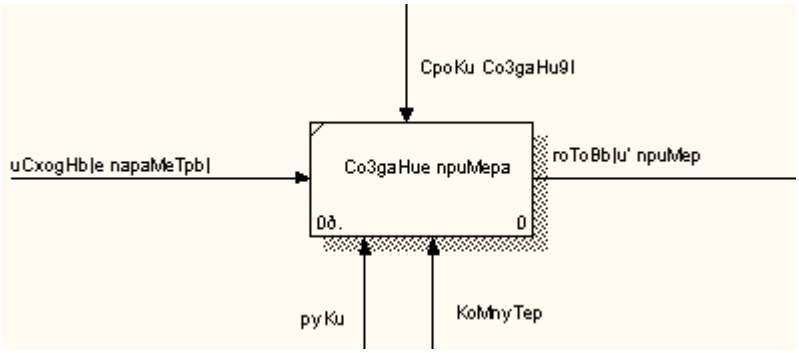


*Рисунок 2*: Интерфейсные дуги

– каждый функциональный блок может быть представлен в виде набора нескольких других функциональных блоков; такое разбиение называется *декомпозиция*; декомпозицию надо понимать как разбиение задач на подзадачи; весь набор декомпозиций блоков создаёт иерархическую схему, которая и будет являться моделью всей системы;
– любой функциональный блок должен иметь вход и выход (соответственно входные слева и выходные справа интерфейсные дуги на формуляре), остальные дуги могут не присутствовать;
– диаграмма, которая включает в себя функциональный блок самого высокого уровня, называется *диаграммой нулевого уровня*;
– диаграмма, содержащая декомпозицию блока диаграммы нулевого уровня называется *диаграммой первого уровня*;
– диаграмма, содержащая декомпозицию блока диаграммы N-го уровня называется диаграммой N+1 уровня;
– любая диаграмма должна содержать (рекомендательно) не менее двух и не более семи функциональных блоков; это необходимо для того, чтобы такие диаграммы были легкочитаемые;
– каждый функциональный блок в рамках единой рассматриваемой системы должен иметь свой уникальный идентификационный номер, названия блоков (пояснения) тоже должны быть уникальны;
– каждая интерфейсная дуга так же должна иметь уникальное имя;

– интерфейсные дуги механизма и управления, используемые в диаграмме N-го уровня также должны входить в диаграмму N-1 уровня; это надо понимать следующим образом: ресурсы необходимые для выполнения подзадачи, считаются необходимыми и для выполнения главной задачи.

Методология IDEF0 имеет большую область применения: от исследования и разработки информационных систем на предприятии, до исследования и разработки схемы работы самого предприятия.

Моделирование производится следующим образом. Сначала строится диаграмма нулевого уровня – она является самым общим описанием системы и показывает границы системы и её взаимодействие с окружающим миром – какие ресурсы внешнего мира она использует, какие результаты своей работы в него отдаёт, какие механизмы и компоненты управления задействуются. Далее производится первая декомпозиция, то есть разбиение задачи на подзадачи. Производится их анализ, и по мере необходимости дальнейшая декомпозиция. Декомпозиция производится до тех пор, пока в этом есть необходимость, то есть детализация задачи недостаточна. Когда задача будет детализирована достаточно, декомпозицию останавливают, и считается, что модель закончена.

Если моделирование начинается на уже существующем предприятии или системе, имеющем уже существующий порядок бизнес-процессов, то с самого начала необходимо произвести анализ работы такого предприятия. В результате анализа получают диаграммы IDEF0, которые наиболее детально описывают работу существующей системы. Такие диаграммы носят название *«AS IS»* (дословно – *«как есть»* с англ.). Детализация и правдивость являются обязательным требованием к диаграммам формата «AS IS». Это связано с тем, что только в этом случае диаграмма может показать реально существующий порядок вещей на предприятии или системе.

Распространённой ошибкой при составлении диаграмм формата «AS IS» является создание идеализированной модели, например руководителем, на основе своих собственных знаний о том, как должны работать сотрудники, а не на основе их знаний. В таком случае модель получается неверной, так как руководитель, как правило, знаком с работой своих сотрудников по должностным инструкциям, не имея при этом представления о настоящем течении работ на предприятии. Такая модель называется *«SHOULD BE»* (дословно – *«как должно было бы быть»* с англ.). Кроме того, обычно сотрудники предприятия, участвующие в системе, заинтересованы в сокрытии истинного положения вещей, поэтому составление модели формата «AS IS» доверяется только приглашённым (сторонним) экспертам.

После составления модели формата «AS IS» производится её анализ. Анализ позволяет выявить недостатки существующего бизнес-процесса, которые могут проявляться, например, в виде недогруженности одних сотрудников одновременно с перегрузкой других. Указание управления и

механизма позволяет выявить идентичные по функционалу задачи, которые удобнее решать в потоке в одном месте и на близких друг от друга местах.

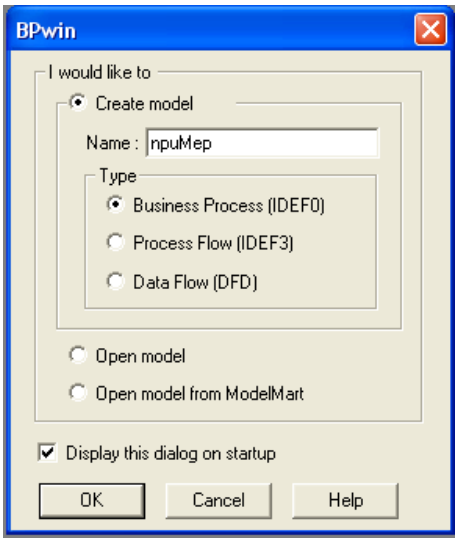
В результате анализа существующего бизнес-процесса или в случае, если производится проектирование бизнес-процесса «с нуля», получают модель формата *«TO BE»* (дословно – *«как должно быть»* с англ.).

Модель «TO BE» свободна от недостатков, которые были свойственны модели «AS IS», что позволяет рассматривать её как идеальную и не всегда достижимую модель бизнес-процессов. Модель «TO BE» можно понимать как оптимизированную модель «AS IS». Так как критерии оптимизации могут быть различны и в общем даже противоположны, то одни и те же блоки модели «AS IS» в зависимости от критериев оптимизации, могут быть в одном случае эффективно работающими, а в другом слабыми местами. Например, критерий высокого качества продукции противоречит критериям дешевизны и высокой производительности, соответственно будут отличаться модели «TO BE».

Для простых бизнес-процессов возможна ситуация, когда модели формата «AS IS» и «TO BE» будут совпадать.

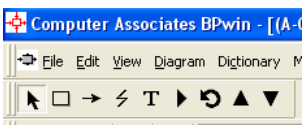
Возможна ситуация, когда модели «AS IS» и «TO BE» различаются незначительно. В этом случае перестройка предприятия или системы под новый режим или схему работы неоправданна.

Ранее, до того как появились персональные компьютеры, диаграммы IDEF0 оформлялись на специальных листах-формулярах. С ростом вычислительных способностей ПК появилась возможность автоматизировать процесс создания диаграмм. Примером программного пакета, позволяющего работать с IDEF0 является пакет BPWin. BPWin является CASE-средством, то есть компьютерной технологией автоматизации.




Работа с IDEF0 в BPWin. При запуске среды появляется приветствие, в котором можно выбрать тип создаваемой модели: IDEF0, IDEF3 или DFD.

После этого появляется диаграмма нулевого уровня в пустым незаполненным функциональным блоком.




### *Задание*

Необходимо составить модель формата «TO BE» по варианту заданному преподавателем. Необходимо составить 1 диаграмму 0го уровня, 1

диаграмму 1го уровня и минимум 2 диаграммы второго уровня, содержащие модели в формате IDEF0.

Варианты заданий:

1. Получение загранпаспорта.
2. Вычисление интеграла.
3. Написание программы.
4. Сдача долгов по экзаменационной сессии.
5. Поступление на работу.
6. Установка операционной системы на компьютер.
7. Процесс обучения по предмету.
8. Выполнение лабораторной (практической) работы
9. Обучение в ВУЗе
10. Бухгалтерский учет
11. Складской учет
12. Бизнес планирование

***Контрольные вопросы.***

1. Что такое бизнес-процесс? Какова его роль на предприятии?
2. Что такое информационная система?
3. Какие типы моделирования информационных систем Вы знаете?
4. Что такое IDEF0?
5. Для чего был разработан стандарт IDEF0?
6. Какие правила используются для нотаций IDEF0?
7. Какие основные определения используются в нотации IDEF0?
8. Какие различают два класса моделей по отношению к реально существующему бизнес-процессу?
9. Что такое CASE-средства, какие CASE средства вы знаете?

## Раздел 2. Разработка интерфейсов пользователя
## МДК.01.02. Разработка интерфейсов пользователя
## Тема 2.1. Разработки прототипов пользовательских интерфейсов

### Практическое занятие № 1. «Знакомство с сервисом figma. Основы работы»

Перед началом выполнения работы необходимо проверить установлено ли все нужное программное обеспечение.

Нужно открыть **Терминал** (MacOS, Linux) или **PowerShell** (Windows) и ввести команды для проверки:

1. NodeJS:

```
node -v
```

2. NPM:

```
npm -v
```

3. Git:

```
git -v
```

Если все команды выдают версии программ, то все в порядке, если же нет, то нужно установить данные программы.

Далее нужно развернуть данный шаблон у себя на компьютере и открыть его в текстовом редакторе Visual Studio Code. Также нужно установить все зависимости для работы проекта, выполнив команду:

```
npm install
```

После этого нужно создать новый репозиторий на github и связать его с проектом, а также сделать первый commit и отправить его в удаленный репозиторий.

Следующим шагом необходимо открыть макет Figma и изучить **StyleKit**:

1. изучить какие шрифты используются в макете;
2. загрузить эти шрифты и поместить их в проект;
3. выгрузить из макета все изображения, иконки и поместить их в проект;
4. изучить и сверстать все отдельные элементы, которые описаны в StyleKit.

Закончив подготовительный этап, можно переходить к верстке.

Чтобы запустить режим разработки для отслеживания всех изменений, нужно открыть терминал в проекте и выполнить команду:

```
npm run dev
```

Первый запуск может быть долгим, в зависимости от конфигурации компьютера, на котором запускается проект. При правильном исполнении сайт будет находиться по адресу https://localhost:8000. Браузер автоматически откроет данную страницу.

Верстку начать лучше всего с семантической разметки header и footer, далее добавить им БЭМ-классы, и только после этого переходить к стилизации.

Далее нужно сверстать все остальные блоки на странице. Блоки верстать лучше по аналогии с header и footer, то есть сначала семантически разметить блок, добавить БЭМ-классы и стилизовать.

Когда вы заканчиваете верстать какой-либо блок, необходимо делать commit и отправлять изменения на удаленный репозиторий. Такой подход предотвратит неприятные ситуации, если ваш код потеряется, не сохранится, случайно удалится и т.д. Вы легко сможете его восстановить из репозитория.

После окончания верстки, для того чтобы создать оптимизированную версию вашего проекта, нужно выполнить команду:

npm run build

При успешном исполнении в проекте появится папка dist внутри которой будут оптимизированные файлы вашей верстки. Как правило, такие файлы передаются backend-разработчику для дальнейшего использования или сразу размещаются на хостинг, если сайт статичный.

**Требования к отчету**

Результатом выполненной работы должен быть подготовленный отчет, который должен соответствовать данной структуре:

1. Титульный лист.
2. Введение.
3. Цели и задачи.
4. Описание выбранных технологий..
5. Ход выполнения работы.
6. Заключение.

Оформление отчета:

- шрифт Times New Roman;
- размер шрифта 14;
- красная строка - 1,25;
- выравнивание текста по ширине;
- межстрочный интервал - 1,5;
- изображения должны быть пронумерованы и подписаны 12 шрифтом, выравнивание по центру;

- листинг программы должен быть вставлен текстом с форматированием как в текстовом редакторе (предварительно включайте светлую тему) и залитый светло-серым цветом.

**Контрольные вопросы**

1. Что такое HTML и для чего его применяют?
2. Что такое семантическая верстка?
3. Что такое CSS и для чего его применяют?
4. Какие виды селекторов бывают?

Что такое псевдоклассы и псевдоэлементы?

## Раздел 4. Работа с системой контроля версий
## МДК.01.03. Тестирование информационных ресурсов

### Тема 4.1. Работа с системой контроля версий.

**Практическая работа №1. Система контроля версий Git.**

*Цель работы:*

Изучить систему контроля версий Git, для чего она нужна и как установить на компьютер.

*Теоретическая часть.*

В настоящее время владение Git стало обязательным требованием при приёме на работу не только для профессионалов, но даже для стажеров.

Git– это система контроля версий (СКВ). Существует несколько подобных систем, однако Git – на настоящий момент наиболее используемая СКВ.

Git создан для решения нескольких проблем любого программиста.

1) История изменений. Работа программиста – это всегда история изменений в коде программы.

**a.** Внося изменения в файлы, хочется знать ответ на вопросы «Кто сделал? Что сделал? Когда сделал?». Таким образом легко отслеживать, когда появились ошибки и кто их сделал. **Любая система, которая позволит нам видеть такую историю изменений, и является системой контроля версий.**

**b.** Иметь возможность отмены изменений или отката по истории назад (и вперёд).

2) Лёгкость внесения изменений.

a. Если нужно попробовать какой-то вариант – это должно быть

просто.

b. Вернуться на основной вариант – также просто

c. Возможность легко принять или отвергнуть альтернативу.

3) Совместная работа

a. Хорошо тому живется, у кого одна нога… Если ног несколько,

возникает проблема синхронизации.

b. Изменения разных пользователей нужно изолировать друг от

друга…

c. … а по мере готовности – сливать вместе.

Программа Git решает все эти проблемы. Эта программа – набор скриптов, который умеет управлять изменениями. Он следит за файлами, ведет их историю, умеет ими манипулировать, откатывать, сливать и т.д.

Git используется в разработке ядра Linux и создан Линусом Торвальдсом.

Задача Git – вести полную историю изменений в некоей папке на сервере или локальном компьютере (**репозиторий**). К изменениям относятся добавление и удаление файлов, модификация их содержимого (нужно также учитывать, что Git не следит за пустыми папками, в папке нужно создать хотя бы один файл).

В историю также записывается, кто и когда сделал изменения.

Git – это распределенная система контроля версий, в ней нет центрального репозитория. Репозиторий – это просто папка с вашими файлами, в которой есть ещё некая служебная информация для Gita. Их может быть очень много. Репозитории могут обмениваться изменениями между собой. Однако, это не DropBox, он не занимается хранением файлов, его задача следующая. Он может от одного репозитория к другому передать изменения!

*Практическая часть.*

**1. Установка Git.**

Для Windows для установки нужно перейти по ссылке.

*git-for-windows.github.io*


Данная версия программы представляет собой не только Git, но и нужное для его работы окружение, которое установится одним пакетом.

Нажимаем Download (вторая кнопка позволяет принять участие в разработке и нам пока не нужна ☺). Устанавливаем всё с настройками по умолчанию.

*Контрольные вопросы:*

1. Что такое СКВ?
2. Какие проблемы решает Git?
3. Для разработки какой операционной системы используется Git?
4. Что такое репозиторий?
5. В чем разница между GoogleDrive/DropBox/YandexDisk и репозиторием Git?

*Список литературы:*

1. Михеева, Е. В. Информационные технологии в профессиональной деятельности :учеб.пособие / Е.В. Михеева. - 14-е изд., стер. - М. : Академия, 2016. - 384 с.

2. Гохберг, Г. С. Информационные технологии : учебник / Г.С. Гохберг, А.В. Зафиевский, А.А. Короткин. - 9-е изд., перераб. и доп. - М. :
Академия, 2014. - 240 с.

## Практическая работа №2 . Создание Git-репозитория.

*Цель работы:*

Научиться создавать новые репозитории в системе контроля Git в среде GitBash. Научиться правильно перемещаться внутри проекта с использованием команд в командной строке, а также установка авторства для проекта, с целью отслеживания изменений.

*Теоретическая часть.*

Система спроектирована как набор программ, специально разработанных с учётом их использования в сценариях. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Например, Cogito является именно таким примером оболочки к репозиториям Git, а StGit использует Git для управления коллекцией исправлений (патчей).

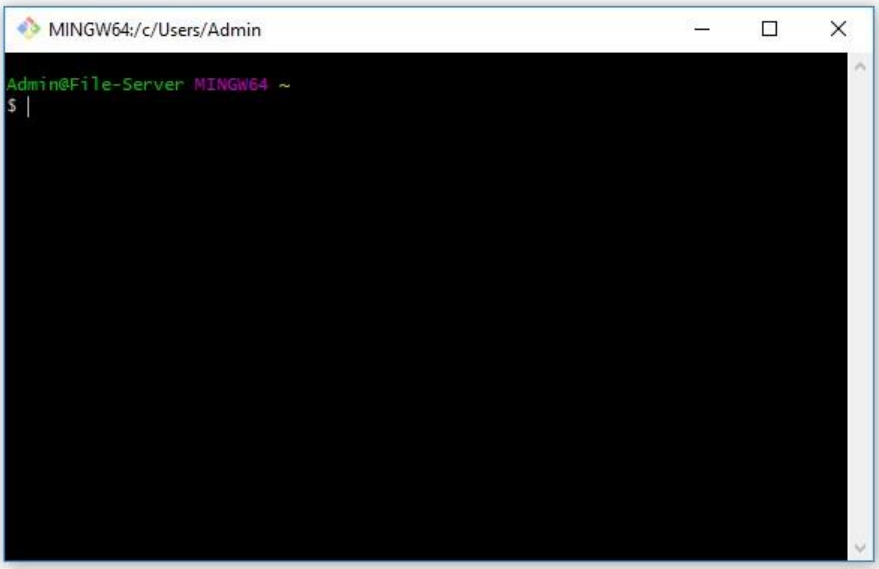
Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Как и Darcs, BitKeeper, Mercurial, Bazaar и Monotone[en], Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается git-демоном, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

*Практическая часть.*

**Создание репозитория.**

2. Запустите GitBash из меню Пуск - Git.
Откроется следующая консоль:

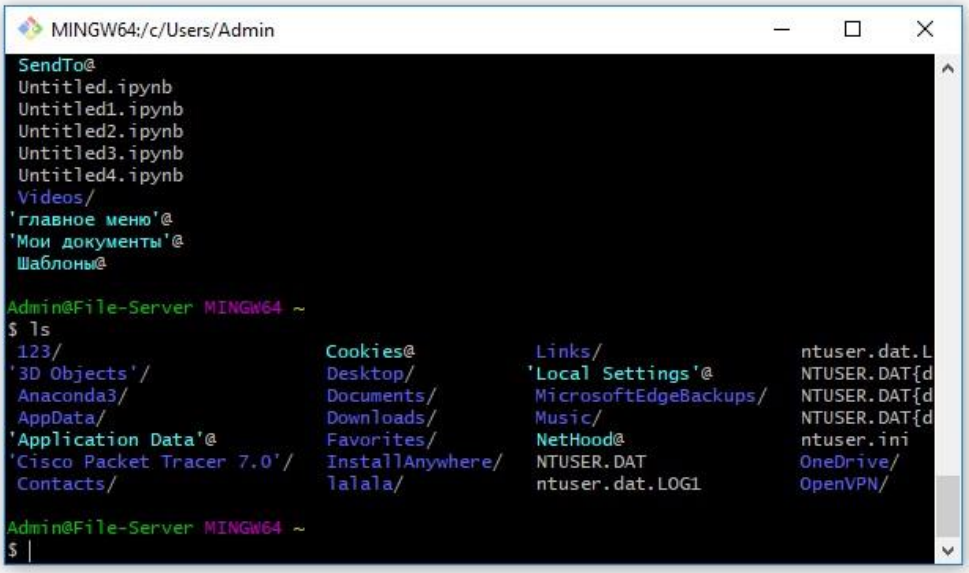



Это командная строка Linux (наподобие консоли командной строки Windows), аккуратно перенесенная в Windows.

Далее работа с Git будет объясняться на примере работы с консольным клиентом по следующим причинам:

- Чтобы у вас складывалось понимание происходящего и при возникновении проблем вы могли четко объяснить, что вы делали, и было видно, что пошло не так.
- Все нажатия кнопок в графических клиентах в итоге сводят к выполнению определенных команд консольного клиента, в то же время возможности графических клиентов ограничены по сравнению с консольным

3. Наберите команду ls. Если после этого вы получите список файлов и папок, значит, Bash успешно установился и запустился.



4. Далее для работы нам потребуется создать папку. Создайте её, например, в корне диска d (непосредственно из-под Windows), назовите TMP.

Теперь нужно в Bash перейти в эту папку. Для этого используем команду cd (changedirectory):

$ cd /d/tmp/

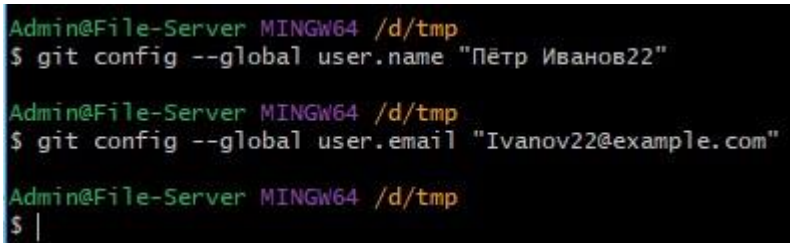
Нажмите Enter и вы окажетесь в этой папке. Если никаких сообщений об ошибке не выводится, значит, команда выполнена правильно.


5. Команда pwd показывает, какая директория текущая в данный момент. Наберите команду и проверьте, где вы находитесь.


6. Далее следует задать настройки Git. Они используются для того, чтобы отслеживать авторов изменений. На своем домашнем компьютере следует задать реальные имя, фамилию и email. В учебной лаборатории задайте имя «Пётр ИвановN», «IvanovN@example.com», где N – номер компьютера в лаборатории (используйте свои имя и фамилию).

Обратите внимание на то, что нажимая кнопку ↑ на клавиатуре, можно повторять ранее использованные команды, они будут выводиться в командной строке, после чего их можно редактировать и выполнять. Это значительно ускорит работу.

Попробуйте после ввода имени повторить команду и отредактировать её, задав адрес электронной почты.




Ключ *–global* означает, что для всех репозиториев будут действовать одни и те же настройки (если задать ключ --local или вообще не задать ключ, настройки будут храниться в данном репозитории и распространяться только на него).

7. Убедитесь, что вы находитесь в папке будущего репозитория (команда *pwd*). Выведите содержимое репозитория (команда *ls*). Убедитесь, что в данный момент папка пуста.

8. Дайте команду

*gitinit*

Эта команда инициализирует репозиторий в текущей пустой папке, о чем выведется сообщение:

```
Admin@File-Server MINGW64 /d/tmp
$ git init
Initialized empty Git repository in D:/tmp/.git/
```

9. Выполните команду ls. Папка по-прежнему пуста.

Теперь введите ту же команду с ключом –a:

```
Admin@File-Server MINGW64 /d/tmp (master)
$ ls

Admin@File-Server MINGW64 /d/tmp (master)
$ ls -a
./  ../  .git/
```

Вы видите, что в папке появились скрытые папки для служебных целей, созданные Git.

10. Попробуйте на диске D: создать пустую папку GitRepo и перенести туда данную папку tmp (сделайте это непосредственно из папки Мой компьютер, Bash можно не использовать). Далее в GitBash перейдите в эту папку (команда cd).

11. Проверьте, что скрытые файлы по-прежнему на месте, т.е. это по-прежнему репозиторий:

```
Admin@File-Server MINGW64 /d/tmp (master)
$ cd /d/GitRepo/tmp/

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ pwd
/d/GitRepo/tmp

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ ls

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ ls -a
./  ../  .git/
```

12. Дайте команду *git status*.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Эта команда показывает, в каком состоянии в данный момент находится наш репозиторий.

В данном случае Git сообщает, что фиксировать нечего, изменений внутри репозитория не было. Т.о. к абсолютному пути репозиторий не привязан.

*Контрольные вопросы:*

1. Что такое GitBash?
2. Для чего нужна команда ls?
3. Как сменить директорию?
4. Как отобразить текущую директорию?
5. Для чего задаются настройки Git?
6. Как применить одинаковые настройки для всех репозиториев?

*Список литературы:*

1. Белов, В. В. Проектирование информационных систем : учебник / В.В. Белов, В.И. Чистяков ; под ред. В.В. Белова. - М. : Академия, 2013. - 352 с. - (Бакалавриат). - На учебнике гриф: Рек.УМО. - Библиогр.: с. 345-347. - ISBN 978-5-7695-7406-1. 2. Хлебников, А. А. Информационные технологии : учебник / А. А. Хлебников. – М. :КноРус, 2014. – 472 с.

**Лабораторная работа №3. Просмотр истории коммитов. Операции отмены.**

*Цель работы:*

Научиться вести историю изменений. Уметь правильно интерпретировать текущий статус репозитория.

*Теоретическая часть.*

Нижний уровень git является так называемой контентно-адресуемой файловой системой. Инструмент командной строки git содержит ряд команд по непосредственной манипуляции этим репозиторием на низком уровне. Эти команды не нужны при нормальной работе с git как с системой контроля версий, но нужны для реализации сложных операций (ремонт повреждённого репозитория и так далее), а также дают возможность создать на базе репозитория git своё приложение.

*Практическая часть.*

**История изменений.**

1. Создайте в папке tmp файл README.txt.
2. Вызовите команду git status.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        README.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Мы видим, что появился не отслеживаемый файл (Untracked files). Т.е. в нашем репозитории появились посторонние файлы, которые Git видит, но пока не отслеживает.

3. Первое, что нужно сделать при таких изменениях – передать файл под контроль Git. Для этого используется команда git add.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git add README.txt
```

Будьте внимательны с регистром!

Если команда ничего не вывела, значит, она завершилась успешно.

4. Проверьте git status снова.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   README.txt
```

На этот раз Git видит этот файл, и сообщает, что он появился в репозитории. Эти изменения ещё не зафиксированы, но они уже проиндексированы. Т.е. Этот файл готов к тому, чтобы быть зафиксированным.

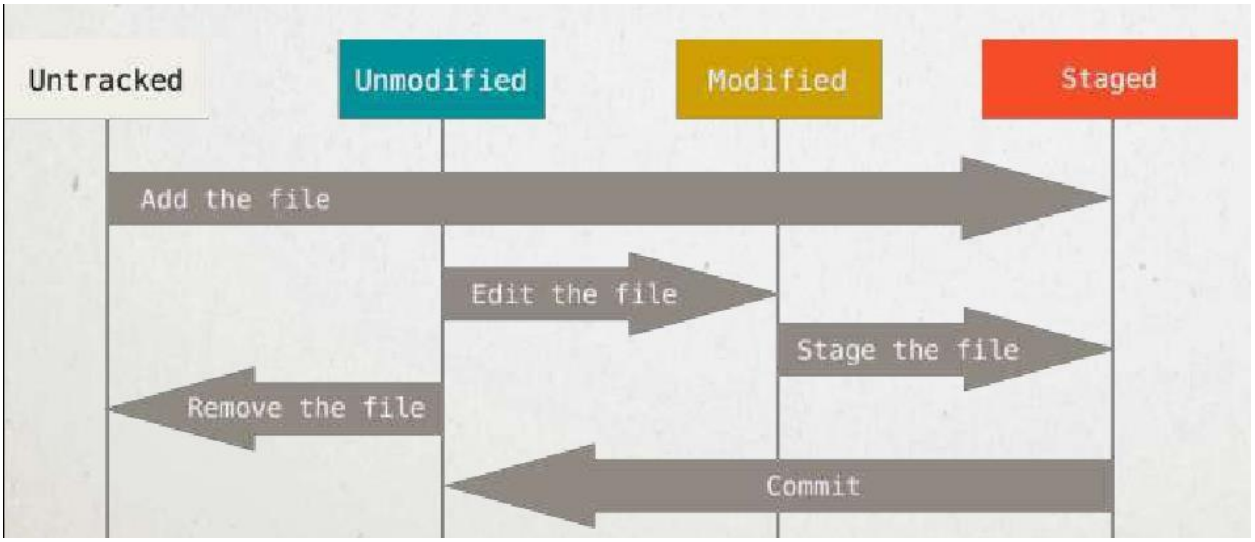
5. Следующий шаг, который мы должны сделать, передав файл под контроль Git и закончив все изменения в нём – закоммитить их (фиксировать изменения).

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git commit -m "Добавлен файл README"
[master (root-commit) 2519868] Добавлен файл README
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.txt
```

Команду gitcommit следует ОБЯЗАТЕЛЬНО использовать с ключом –m и комментарием!

Рассмотрим полученное от Git сообщение. Тут написано, что один файл изменён.

Итак, в Git файлы могут находиться в четырёх состояниях:




1) Untracked – файл просто находится в папке, но для Git он неизвестен, его в истории нет и никогда не было.

2) Staged – подготовленный для фиксации изменений (командой add). Это значит, что файл под контролем Git и он ждёт нашей команды, чтобы зафиксировать изменения (т.е. установить флажок в истории с комментарием).

3) Unmodified–означает, что файл со времени последней фиксации не менялся. Переводится в это состояние фиксацией (commit).

4) Modified – измененный файл. Этот файл уже был в истории, Git о нём знает, файл участвовал в каких-то коммитах, но мы его сейчас изменили. Теперь мы можем либо перевести его в Staged (add), либо отменить изменения.

Наша задача – отчётливо понимать, как происходит перемещение файла по этим состояниям. Результатом работы должно быть состояние файлов Unmodified.

5. Проверьте gitstatus снова. В данный момент все должно быть Unmodified – nothingtocommit.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master
nothing to commit, working tree clean
```

6. Откройте файл README.txt и напишите в нем «Hello, world!». Сохраните файл. Проверьте gitstatus снова.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Git сообщает о том, что файл был изменён. И он не готов к фиксации (changes not staged for commit).

Каким образом Git это определяет? У него в папке хранятся идеальные копии наших файлов, и он очень быстро их сравнивает с реальными файлами.

7. Готовим изменения к фиксации: *gitaddREADME.txt*

8. Проверяем текущий статус.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git add README.txt

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   README.txt
```

Видим, что измененный файл *README.txt* готов к фиксации.

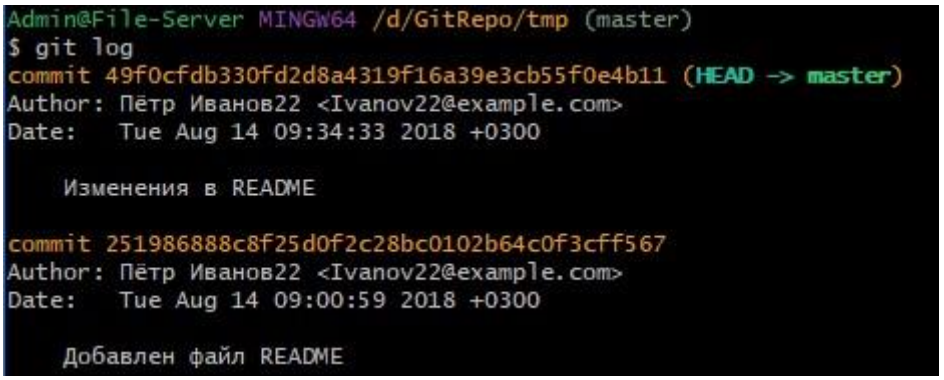
9. Далее следует зафиксировать изменения (commit). Не забудьте про ключ и комментарий!!!

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git commit -m "Изменения в README"
[master 49f0cfd] Изменения в README
 1 file changed, 1 insertion(+)
```

Дополнительные команды:

*gitrmFILE*–удаляет файл из индекса и из рабочей папки; *gitrm --cachedFILE* – удаляет файл только из индекса Git. Оба этих удаления требуют фиксации!

10. Для того, чтобы просмотреть историю коммитов, нужно использовать команду gitlog – это лог всех изменений.

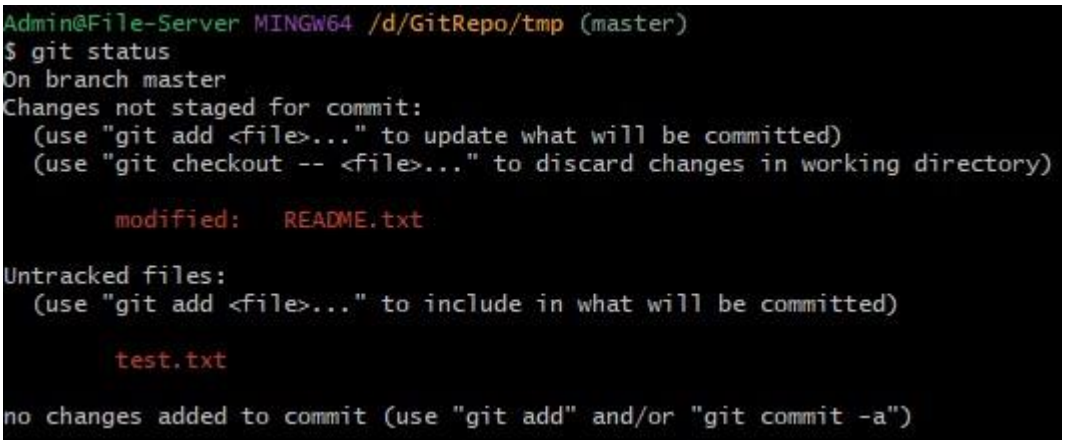



Здесь мы видим созданные нами два коммита. Длинные числа из шестнадцатеричных цифр – это их номера (кэш). Такие длинные номера нужны для того, чтобы они были уникальными. Они непоследовательны, это сделано специально, для веток (они будут рассмотрены далее).

К каждому коммиту указан автор и дата/время изменений.

11. Создайте в нашей папке ещё один файл test.txt, напишите внутри него TEST. В файле README добавьте пару восклицательных знаков или что-либо ещё. Таким образом мы получили два изменения в репозитории.

12. Проверьте текущий статус.




Мы видим сообщение о том, что два изменения дожидаются **стейджинга**, т.е. индексации. Для чего вообще выполняется индексация? Таким образом мы показываем, что изменения в данном файле важны и мы планируем их зафиксировать. Если файл не проиндексирован, значит, мы продолжаем его обрабатывать и не уверены, что будем сохранять изменения.

13. Добавляем (индексируем) оба файла:

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git add README.txt

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git add test.txt

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   README.txt
        new file:   test.txt
```

14. Зафиксируем оба изменения в одном коммите:

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git commit -m "Тестовый коммит с 2 изменениями"
[master 57f6580] Тестовый коммит с 2 изменениями
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 test.txt
```

Просмотрите историю коммитов в gitlog. Там появился новый только что созданный коммит.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git log
commit 57f658087d95ad4feb553ef3f30309422abba458 (HEAD -> master)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Aug 14 12:07:40 2018 +0300

    Тестовый коммит с 2 изменениями

commit 49f0cfdb330fd2d8a4319f16a39e3cb55f0e4b11
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Aug 14 09:34:33 2018 +0300

    Изменения в README

commit 251986888c8f25d0f2c28bc0102b64c0f3cff567
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Aug 14 09:00:59 2018 +0300

    Добавлен файл README
```

Чтобы более подробно видеть информацию о коммитах, в gitlog можно использовать ключ -p. В этом случае по каждому изменению будет выведено, что было и что стало в каждом измененном файле.

**Отмена коммитов.**

Крайне нежелательно отменять коммиты, особенно при работе в группе. Это похоже на бухгалтерскую проводку – если документ был проведён, то его нельзя удалить, а можно лишь выполнить коррекцию. Однако, если очень нужно, то для этого есть специальные команды. gitresetHEAD– откат к последнему коммиту (здесь HEAD– указатель на текущий коммит). То есть, если что-то пошло не так, то

можно отменить все изменения и откатиться к состоянию после последнего коммита.

gitreset --hard<commit> - самая мощная и самая опасная команда – откат к указанному коммиту с потерей всех изменений!

1. Для примера что-нибудь изменим, а затем откатимся к состоянию после коммита. Откройте файл README, замените !!! на ???. Файл test.txt удалите.

2. Выведите текущий статус.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   README.txt
        deleted:    test.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Далее выполните gitadddля обоих файлов, чтобы уж окончательно проиндексировать изменения.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git add README.txt

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git add test.txt

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   README.txt
        deleted:    test.txt
```

И вдруг в этот самый последний момент мы решаем всё изменить!

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git reset HEAD
Unstaged changes after reset:
M       README.txt
D       test.txt
```

Всё вернулось в состояние до индексации (убедитесь в этом, проверив текущий статус).

3. Далее, чтобы вернуться в состояние до изменений, следует использовать команду gitcheckout--test.txtи gitcheckout--README.txt.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git checkout test.txt

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git checkout README.txt

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Убедитесь, что удалённый файл вернулся на место, а во втором отменились изменения.

4. Команда gitreset --hard<commit>, в отличие от предыдущей, работает не по шагам, а разом. Рассмотрим её действие. Удалите и исправьте файлы, как в пункте 1.

Затем примените команду gitreset --hard и проверьте полученный статус:

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git reset --hard HEAD
HEAD is now at 57f6580 Тестовый коммит с 2 изменениями

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Всё вернулось на места к состоянию последнего коммита.

5. Попробуем откатиться не на последний, а на предпоследний коммит. Для этого нужно указать его номер (несколько цифр, позволяющих идентифицировать его однозначно). Чтобы знать номера коммитов, сначала выведите лог. Затем выполните откат по номеру предпоследнего коммита:

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git log
commit 57f658087d95ad4feb553ef3f30309422abba458 (HEAD -> master)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Aug 14 12:07:40 2018 +0300

    Тестовый коммит с 2 изменениями

commit 49f0cfdb330fd2d8a4319f16a39e3cb55f0e4b11
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Aug 14 09:34:33 2018 +0300

    Изменения в README

commit 251986888c8f25d0f2c28bc0102b64c0f3cff567
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Aug 14 09:00:59 2018 +0300

    Добавлен файл README

Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ git reset --hard 49f0cfdb
HEAD is now at 49f0cfd Изменения в README
```

Проверьте папку, убедитесь, что всё вернулось к состоянию предпоследнего коммита (второй файл исчез, а в первом только один восклицательный знак).

Примечание:

Если лог не помещается на экране, то программа будет выводить его по частям и после первой части ждать нажатия любой клавиши. Вверх-вниз его можно проматывать стрелками на клавиатуре. Для выхода из этого режима следует нажать «q».

**Самостоятельная работа**

1. Создайте репозиторий в пустой папке
2. Добавьте в папку файл. Изучите вывод команды git status. Проиндексируйте файл командой git add. Снова посмотрите вывод git status.
3. Зафиксируйте изменения командой git commit.
4. Сделайте и зафиксируйте следующие изменения (каждый подпункт - одна фиксация):
   - Добавление сразу трех файлов
   - Изменения в тексте двух файлов
   - Изменения в тексте одного файла, удаление другого и добавление еще одного

5.* Удалите из какой-либо подпапки все файлы и зафиксируйте это изменение (подпапку, конечно же, надо предварительно создать). А затем удалите и саму пустую подпапку. Объясните результат.

6. Изучите вывод команды git log. Прочтите о ее различных ключах:

https://git-scm.com/book/ru/v1/%D0%9E%D1%81%D0%BD%D0%BE%D0%B2%D1%8B-Git-%D0%9F%D1%80%D0%BE%D1%81%D0%BC%D0%BE%D1%82%D1%80-%D0%B8%D1%81%D1%82%D0%BE%D1%80%D0%B8%D0%B8-%D0%BA%D0%BE%D0%BC%D0%BC%D0%B8%D1%82%D0%BE%D0%B2

7. * Создайте свой формат вывода git log с помощью ключа --

pretty=format

8. * Внесите изменения в репозиторий, но не фиксируйте их. Посмотрите git status. Теперь дайте команду git stash и снова посмотрите статус и свои файлы. А теперь git stash apply. Попробуйте объяснить результат, не прибегая к документации.

В отчёт приложите все использованные вами команды и их вывод в консоль.

В отчёте опишите, как вы поняли принцип действия всех новых команд из самостоятельной работы.

Задания, помеченные знаком "*" требуют самостоятельного поиска дополнительных материалов.

*Контрольные вопросы:*

1. Как отобразить текущий статус репозитория?
2. Какие файлы называются Untracked files?
3. Как передать файл под контроль Git?
4. Как отобразить список изменений в репозитории?
5. Как отменить коммит и почему это делать не рекомендуется?
6. Как Git идентифицирует все этапы изменения?

*Список литературы:*

1. Белов, В. В. Проектирование информационных систем : учебник / В.В. Белов, В.И. Чистяков ; под ред. В.В. Белова. - М. : Академия, 2013. - 352 с. - (Бакалавриат). - На учебнике гриф: Рек.УМО. - Библиогр.: с. 345-347. - ISBN 978-5-7695-7406-1. 2. Хлебников, А. А. Информационные технологии : учебник / А. А.

Хлебников. – М. :КноРус, 2014. – 472 с.

**Практическая работа №4. Работа с удалёнными репозиториями**

*Цель работы:*

Научиться работать с удаленными репозиториями. Загрузка веток из других источников. Операции с удаленными ветками

*Теоретическая часть.*

Удалённые ветки — это ссылки на состояние веток в ваших удалённых репозиториях. Это локальные ветки, которые нельзя перемещать; они двигаются автоматически всякий раз, когда вы осуществляете связь по сети. Удалённые ветки действуют как закладки для напоминания о том, где ветки в удалённых репозиториях находились во время последнего подключения к ним.

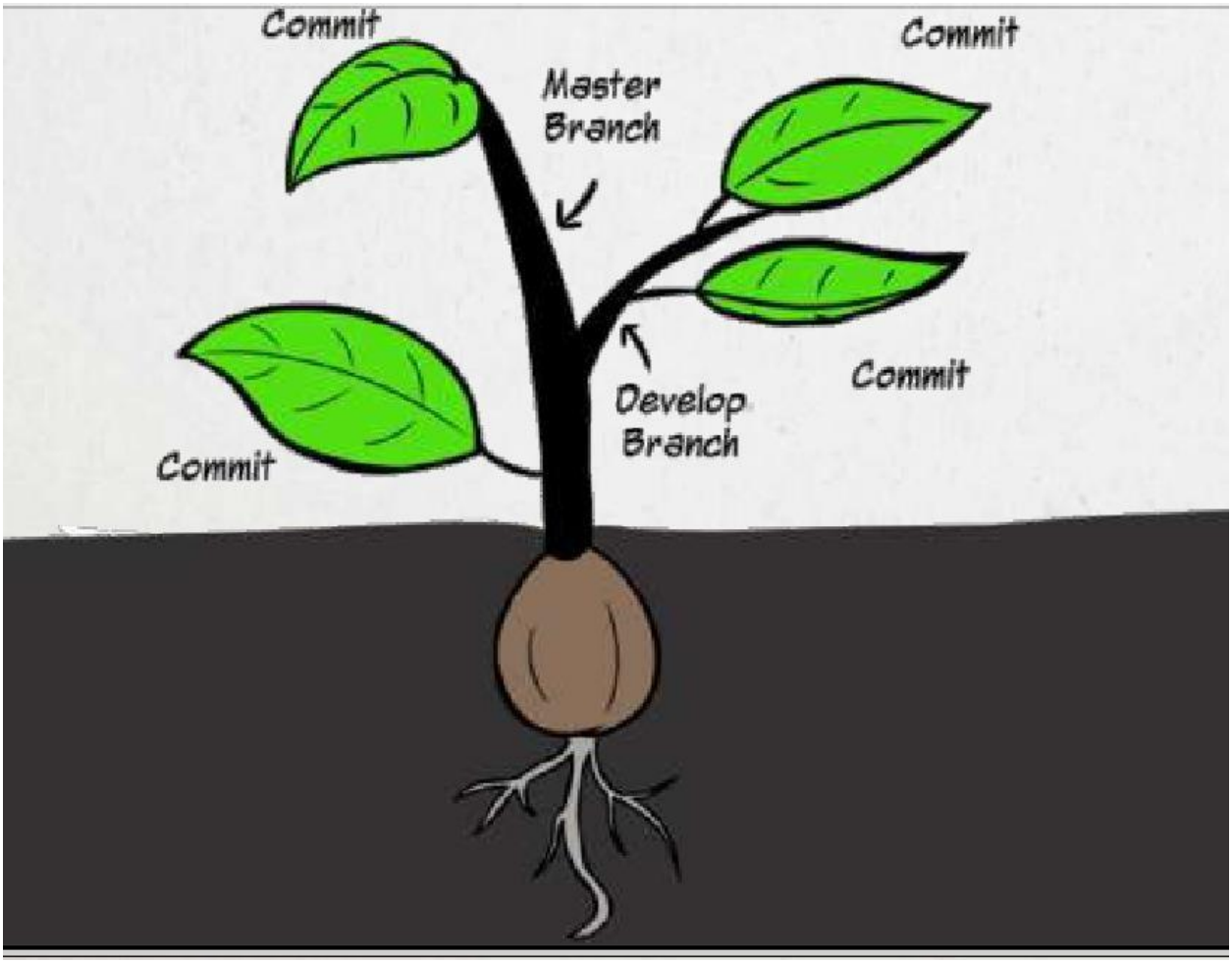
Они выглядят как (имя удал. репоз.)/(ветка). Например, если вы хотите посмотреть, как выглядела ветка master на сервере origin во время последнего соединения с ним, проверьте ветку origin/master. Если вы с партнёром работали над одной проблемой, и он выложил веткуiss53, у вас может быть своя локальная ветка iss53; но та ветка на сервере будет указывать на коммит в origin/iss53.

*Практическая часть.*

Когда на экран выводится git log, мы видим слово (master). Это – основная ветвь истории изменений. На данный момент она для нас единственная.

**Ветка** – это последовательность коммитов.

1. Дайте команду git status в своём репозитории. Будет выведено сообщение «On branch master» - «Я нахожусь на ветке master» - в главной ветви, которая начинается с самого первого коммита в истории и доходит до последнего.




В данной работе мы будем рассматривать только ветвь master.

Git – это распределенная система, это значит, что у нас может быть много репозиториев. Они могут быть распределены – находиться на разных компьютерах внутри локальной сети, в разных местах одного компьютера или где-то далеко в интернете. Распределенность Git заключается в том, что он умеет определенным образом связывать эти репозитории между собой.

В некоторых СКВ система репозиториев клиент-серверная, но не в Git. Здесь все репозитории равноправны между собой.

Репозиотрии делятся ан 2 типа – локальные (у нас на компьютере) и удалённые (где-то ещё). Git умеет связывать репозитории между собой. Цель этой лабораторной работы – изучить данную технологию, то, как эта связь выполняется.

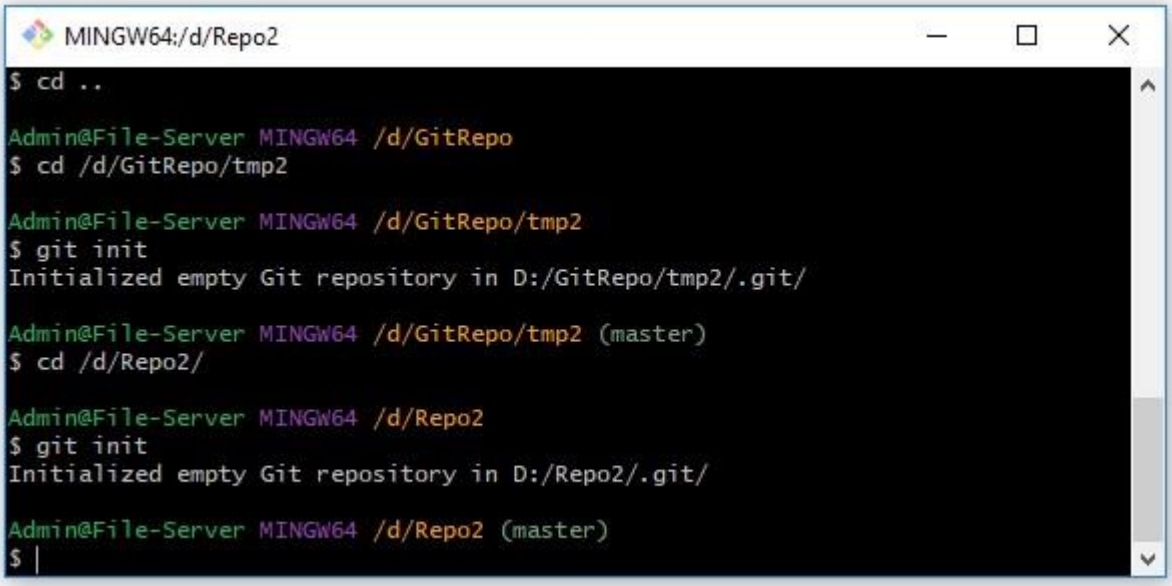
Для работы с удаленными репозиториями используется команда git remote.

2. Создадим в папке GitRepo новый чистый репозиторий tmp2, а старый удалим. В Git Bash перейдем в новую папку (cd <путь>).

```
Admin@File-Server MINGW64 /d/GitRepo/tmp (master)
$ cd ..

Admin@File-Server MINGW64 /d/GitRepo
$ cd /d/GitRepo/tmp2

Admin@File-Server MINGW64 /d/GitRepo/tmp2
$
```

Инициализируйте репозиторий (git init).

3. Создайте аналогично второй репозиторий в корне диска D:




Добавьте в него файл README1.txt с каким-либо текстом, сделайте коммит. Затем в Git Bash сделайте текущим первый репозиторий.

```
Admin@File-Server MINGW64 /d/Repo2 (master)
$ git add README1.txt

Admin@File-Server MINGW64 /d/Repo2 (master)
$ git commit -m "Новый файл в Repo2"
[master (root-commit) 928ae36] Новый файл в Repo2
 1 file changed, 1 insertion(+)
 create mode 100644 README1.txt

Admin@File-Server MINGW64 /d/Repo2 (master)
$ cd /d/GitRepo/tmp2/
```

4. Команда *git remote add* добавляет к текущему репозиторию ссылку на удалённый (находящийся в другом месте) под каким-то именем, назовём его super2.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp2 (master)
$ git remote add super2 "d:\Repo2"
```

5. Команда git remote покажет нам список прикрепленных к нашему репозиторию удалённых репозиториев.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp2 (master)
$ git remote
super2
```

С ключом –v вывод будет немного подробнее:

```
Admin@File-Server MINGW64 /d/GitRepo/tmp2 (master)
$ git remote -v
super2  d:\Repo2 (fetch)
super2  d:\Repo2 (push)
```

Показывается не только название, но ещё и адрес удалённого репозитория.

Связь создана, однако на статус это никак не повлияет.

Для просмотра удаленного репозитория можно использовать команду *git remote show super2*

6. Далее выполним команду git fetch super2

Эта команда возьмет информацию обо всех изменениях в удалённом репозитории и перенесёт в текущий.

Обратите внимание, что будет перенесена только информация об изменениях, а не они сами!!! fetch не меняет файлы!!!

```
Admin@File-Server MINGW64 /d/GitRepo/tmp2 (master)
$ git fetch super2
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From d:\Repo2
 * [new branch]      master     -> super2/master
```

Выведите содержимое репозитория командой ls. Вы увидите, что он не изменился. В нем по-прежнему ничего нет. Однако, теперь мы знаем всё об удалённом репозитории.

После того, как мы сделали git fetch, мы получили в том числе список веток удалённого репозитория. В данный момент у нас есть одна ветка – master. А в удаленном репозитории может быть много разных веток. Для того, чтобы работать с удаленными изменениями, нам нужно связать ветки между собой. В самом простом случае – связать две ветки master между собой.

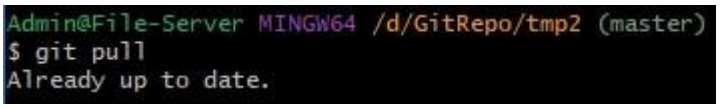
7. Выполните команду *git checkout --track super2/master*

Здесь сказано, что Git должен создать связь текущей ветки данного репозитория с ветвью master репозитория super2.

```
Admin@File-Server MINGW64 /d/GitRepo/tmp2 (master)
$ git checkout --track super2/master
Already on 'master'
Branch 'master' set up to track remote branch 'master' from 'super2'.
```

В ответ Git пишет, что ветка master установлена для отслеживания удалённой ветки master из super2.

8. Команда git pull – команда, которая получает изменения.




Т.е. Эта команда позволяет выкачать все изменения из удаленного репозитория.

В следующей работе рассмотрим использование для этих целей популярного в настоящее время сервиса Github.

*Контрольные вопросы:*

1. Что такое ветка?
2. На какие типы делятся репозитории Git?
3. Как добавить ссылку на удаленный репозиторий?
4. Как вывести подробный список прикрепленных репозиториев?
5. Что выполняет команда *git remote show*?

*Список литературы:*

1. Хлебников, А. А. Информационные технологии : учебник / А. А. Хлебников. – М. :КноРус, 2014. – 472 с.
2. Золотов, С.Ю. Проектирование информационных систем : учебное пособие / С.Ю. Золотов ; Министерство образования и науки Российской Федерации, Томский Государственный Университет

Систем Управления и Радиоэлектроники (ТУСУР). - Томск : Эль Контент, 2013. - 88 с. : табл., схем. - ISBN 978-5-4332-0083-8 ; То же

[Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=208706

**Практическая работа №5. Работа с метками.**

*Цель работы:*

Научиться присваивать метки к репозиториям. Работа с сабмодулями

*Теоретическая часть.*

Как и большинство СКВ, Git имеет возможность помечать (tag) определённые моменты в истории как важные. Как правило, этот функционал используется для отметки моментов выпуска версий (v1.0, и т.п.). В этом разделе вы узнаете, как посмотреть имеющиеся метки (tag), как создать новые. А также вы узнаете, что из себя представляют разные типы меток.

Просмотр имеющихся меток (tag) в Git'е делается просто. Достаточно набрать git tag.

Git использует два основных типа меток: легковесные и аннотированные. Легковесная метка — это что-то весьма похожее на ветку, которая не меняется — это просто указатель на определённый коммит. А вот аннотированные метки хранятся в базе данных Git'а как полноценные объекты. Они имеют контрольную сумму, содержат имя поставившего метку, e-mail и дату, имеют комментарий и могут быть подписаны и проверены с помощью GNU Privacy Guard (GPG). Обычно рекомендуется создавать аннотированные метки, чтобы иметь всю перечисленную информацию; но если вы хотите сделать временную метку или по какой-то причине не хотите сохранять остальную информацию, то для этого годятся и легковесные метки.

*Практическая часть.*

**Теги**

Теги – это метки, которыми можно пометить коммит. Они часто применяются в работе.

Есть лёгкие теги. Они очень простые. Мы берем команду gittagи придумываем какую-либо метку. Часто теги используют, чтобы помечать какие-либо версии. Например, показать, что на каком-то коммите мы достигли версии 1.0. То есть, это просто ещё одно имя для коммита, для того, чтобы пометить какие-то важные в истории коммиты.

1. Введите команду gittag– это команда, которая выводит состояние меток в текущем репозитории. В данном случае меток нет, ничего не будет выведено.
2. Вызовите команду gitstatus, убедитесь, что вы находитесь в ветке master. Теперь пометим тегом 1.0 текущий коммит.

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git status
On branch master
nothing to commit, working tree clean

Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git tag 1.0
```

3. Повторите команду gittag. Вы увидите, что у вас появился тег.

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git tag
1.0
```

4. Если вы хотите увидеть помеченный коммит, используйте команду gitshow.

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git show 1.0
commit ecd8cd72b1ebaf1946bc89ac90c50223d3065971 (HEAD -> ma
re/1)
Merge: fe36a38 2374d7c
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Oct 2 14:31:58 2018 +0300

    Конфликт решен

diff --cc first.txt
index 5ace0b5,cdc9555..51bd26d
--- a/first.txt
+++ b/first.txt
@@@ -1,3 -1,3 +1,3 @@@
  Это первая строчка.

- Красная кнопка
 -Тут две зелёных кнопки.
++Одна красная и две зеленые кнопки.
```

Вы видите, что в этом коммите был решен конфликт – проблема с красной и зелеными кнопками.

5. Меток в проекте может быть множество, поэтому у команды git tag может быть метка -l – маска. Например:

git tag -l ‘1.*’

Эта маска покажет, какие метки начинаются с «1.»

6. Для удаления тега применяется команда gittag -d. Например: git tag -d 1.0

Кроме лёгких тегов существуют **аннотированные метки**. Это тоже метка (имя для коммита), однако, кроме имени в ней можно поместить некоторое сообщение, дату и время создания, имя того, кто создал, и даже подписать цифровой подписью.

Это достаточно сложная возможность, она используется в больших проектах, где много разработчиков сливают свои изменения, и есть один главный разработчик, которому поручено принимать решения о выпуске продукта. Они используются для того, чтобы

пометить коммит готового к сборке продукта. Этот человек (релиз-мастер) с помощью своей цифровой подписи свидетельствует о том, что он даёт визу на публикацию. Он ставит метку на этот коммит и подписывает её. Автоматические системы сборки на это реагируют и начинают сборку продукта (эти технологии не слишком используются обычными разработчиками). Выглядит эта команда так: git teg -a 1.0 – m “Тест тега”

7. Создайте такой тег.

8. Используйте команду git tag 1.0. В списке тегов вы увидите этот тег в обычном виде.

9. Вызовите команду git show 1.0. Вы увидите намного больше информации об этом коммите, чем с обычным тегом, в том числе (сразу после команды) сведения о том, кто эту метку поставил (строчка Tagger), когда он её поставил (строчка Date). Если бы мы воспользовались при создании тега своим ключом, то тут бы была также информация о нём, и некоторый автоматизированный софт смог бы проверить её подлинность. Это используется в очень крупных проектах, например, при работе над Линуксом.

Метки можно создавать и позже. Достаточно при создании указать номер уже существующего коммита:

git tag -a 1.0 -m "Сообщение" 1baf

10. Выведите список коммитов.

11. Создайте тег 1.1 для созданного ранее коммита.

Метки по умолчанию не передаются в удаленный репозиторий. Для передачи используйте команды:

git push origin 1.0 – ДЛЯ КОНКРЕТНОЙ МЕТКИ git push origin --tags - ДЛЯ ВСЕХ ТЕГОВ

Это происходит, так как git push отправляет изменения, а метка изменением не является.

**СУБМОДУЛИ**

СУБМОДУЛЬ – это, фактически, репозиторий внутри вашего репозитория.

**git submodule add РЕПОЗИТОРИЙ ПАПКА**

Для чего он может быть нужен? Например, вы работаете над проектом и нашли какую-то библиотеку, которую хотите к нему подключить (например, нашли её на github). Однако, её нельзя просто склонировать на ваш компьютер, перетащить файлы к себе и закоммитить. Однако, в этом случае, если автор внесёт какие-то коррективы в свою библиотеку, вы об этом никогда не узнаете.

Хорошим подходом в этом случае является субмодуль. Т.е. Можно просто включить в ваш репозиторий ссылку на другой.

1. Создайте на hithub под своим профилем второй репозиторий, добавьте в него файлы и включите его в ваш текущий проект. Для этого после того, как скопировали ссылку на репозиторий, перейдите в Git Bash в текущий репозиторий и выполните команду git submodule add ____вставить адрес___ app

(app – название нового репозитория)

2. Откройте папку с репозиторием и убедитесь, что там появилась новая папка с субмодулем.

3. Проверьте git status. Вы видите, что добавился автоматически файл gitmodules, который описывает все подключенный модули, его можно прочитать (cat .gitmodules). Также добавилась папка app, которая в статусе отображается как одно целое, хотя внутри она содержит другие файлы и папки библиотеки.

4. Сделайте коммит и у вас зафиксируется информация о том, что в папке app находится клон удалённого репозитория.

После добавления субмодуля нужно зафиксировать изменения, обычной командой git commit

5. Сделайте коммит.

Есть ещё две необходимые команды для работы с субмодулями:

**git submodule init** инициализирует субмодули в репозитории, где это еще не сделано (например сразу после клонирования) **git submodule update**

• Загружает файлы субмодулей

*Контрольные вопросы:*

1. Что такое метка?
2. Какие типы меток существует?
3. Как добавляется метка к репозиторию?

4. Как вывести информацию по метке?
5. Для чего нужен сабмодуль?
6. Как добавить сабмодуль?

*Список литературы:*

1. Белов, В. В. Проектирование информационных систем : учебник / В.В. Белов, В.И. Чистяков ; под ред. В.В. Белова. - М. : Академия, 2013. -

352 с. - (Бакалавриат). - На учебнике гриф: Рек.УМО. - Библиогр.: с. 345-347. - ISBN 978-5-7695-7406-1. 2. Хлебников, А. А. Информационные технологии : учебник / А. А.

Хлебников. – М. :КноРус, 2014. – 472 с.

**Практическая работа №6. Работа с ветками**

*Цель работы:*

Теперь, когда вы уже попробовали создавать, объединять и удалять ветки, пора познакомиться с некоторыми инструментами для управления ветками, которые вам пригодятся, когда вы начнёте использовать ветки постоянно.

*Теоретическая часть.*

Команда git branch делает несколько больше, чем просто создаёт и удаляет ветки. При запуске без параметров, вы получите простой список имеющихся у вас веток: $ git branch iss53 * master testing

Обратите внимание на символ *, стоящий перед веткой master: он указывает на ветку, на которой вы находитесь в настоящий момент (т.е. ветку, на которую указывает HEAD). Это означает, что если вы сейчас выполните коммит, ветка master переместится вперёд в соответствии с вашими последними изменениями. Чтобы посмотреть последний коммит на каждой из веток, выполните команду git branch -v:

```
$ git branch -v iss53
93b412c fix javascript issue *
master 7a98805 Merge branch
'iss53'
  testing 782fd34 add scott to the author list in the readmes
```

*Практическая часть.*

***Понятие «Ветки».***

Репозиторий хранит в себе информацию о состояниях репозитория. Они называются «снапшоты». Каждое состояние – это коммит. Коммит – это ***снапшот и данные об авторстве, времени и номере коммита, а также указатель на родительский коммит***.Снапшот – это информация о том, каким был репозиторий в какой-то момент времени. Это полная информация обо всех файлах и папках, снимок файловой системы. Он тщательно упакован и занимает минимально возможное место.

В отличие от других систем контроля версий, это не запись об изменениях от предыдущего состояния, а вся информация о репозитории. Это позволяет быстро перемещаться по коммитам, не внося изменения в несколько этапов, а просто перейдя на нужный коммит.

Это также позволяет в случае какой-то потери данных пропустить битый коммит и перейти к предыдущей рабочей версии.

Коммит снабжен указателями на его родительские коммиты:

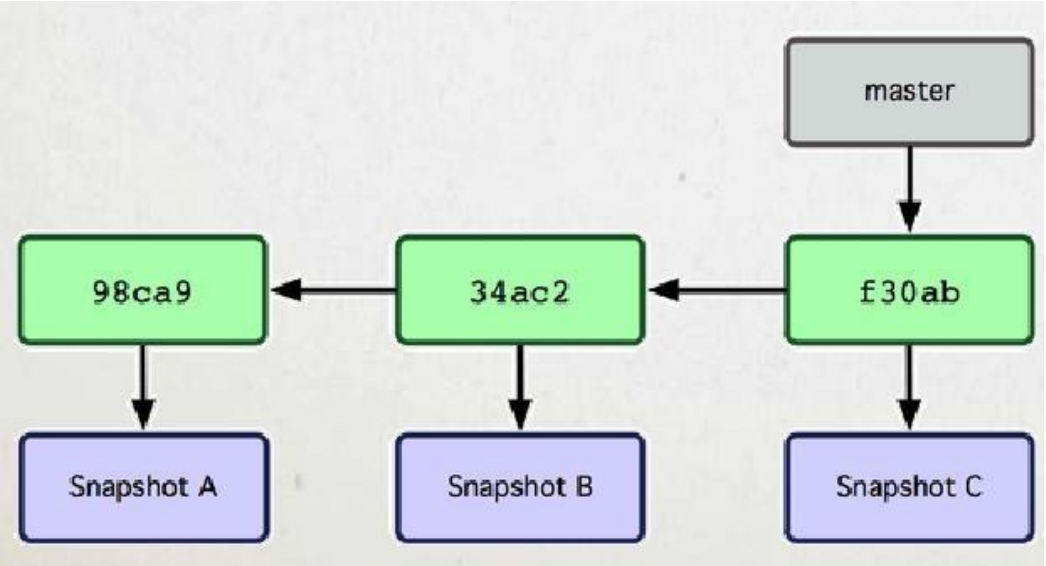
- Ноль указателей – если это первый коммит;
- Один – если это обычный;
- Несколько – в случае слияния изменений (об этом пойдет речь позже)

Таким образом, система всех этих указателей на родительские коммиты создает историю проекта, где в самом конце стоит самый первый коммит.

***Ветка*** – это указатель на какой-либо коммит в истории.

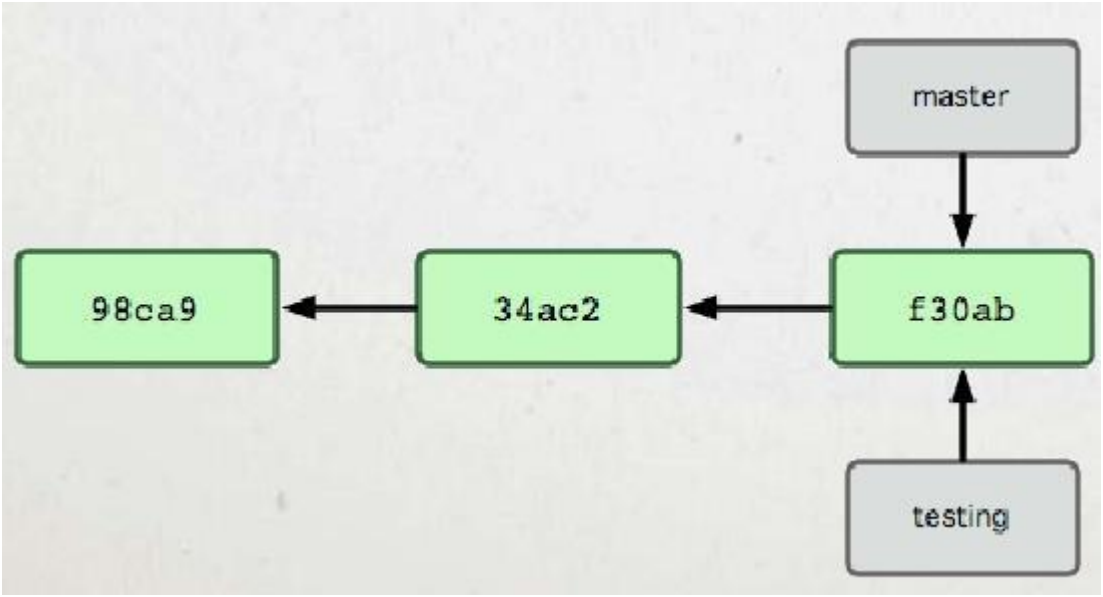
При создании репозитория автоматически создается ветка master. Указатель на нее автоматически сдвигается при каждом коммите. Ветка master считается основной, хотя это просто договорённость.

На рисунке представлено перемещение указателя «master». Сначала был создан снапшот A и указатель мастер был на нем, потом создали снапшотB, указатель master переместился на него, затем перешел на станпшот C:




Репозитории в Git могут содержать неограниченное число новых веток. Для того, чтобы создать новую ветку, нужно дать команду gitbrunch<новая ветка>.

Например, команда gitbrunchtesting создаст новую ветку testing, то есть создаст новый указатель на текущее состояние репозитория и даст ему имя testing. Больше эта команда ничего не делает! В результате история будет выглядеть следующим образом:




Таким образом, на один и тот же коммит могут указывать сколько угодно веток.

Перейдем к практике.

1. Создайте на диске d папку ProGit и в ней папку brunches:

```
Танюшка@NoteBook MINGW64 ~
 cd /d/

Танюшка@NoteBook MINGW64 /d
 mkdir ProGit

Танюшка@NoteBook MINGW64 /d
 mkdir /d/ProGit/branches/

Танюшка@NoteBook MINGW64 /d
 cd /d/ProGit/branches/

Танюшка@NoteBook MINGW64 /d/ProGit/branches
```

2. Инициализируйте репозиторий в этой папке

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git init
Initialized empty Git repository in D:/ProGit/branches/.git/
```

3. Проверьте статус репозитория, убедитесь, что он совершенно пуст. Мы находимся на ветке master (Onbranchmaster).

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

4. Создайте из-под Windows в этой папке какой-либо текстовый файл (например, README.txt), напишите в него текст «Проверка веток».

5. Добавьте этот файл и сохраните коммит под названием «Первый коммит».

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git add ./README.txt

Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git commit -m "Первый коммит"
[master (root-commit) 05b133a] Первый коммит
 1 file changed, 1 insertion(+)
 create mode 100644 README.txt
```

6. Проверьте статус репозитория:

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git status
On branch master
nothing to commit, working tree clean
```

Мы по-прежнему находимся в ветви master.

7. Вызовите git log, и вы увидите, что существует только один «Первый коммит».

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git log
commit 05b133ad74ed028634f770e1dd94067aae6d9cbb (HEAD -> master)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Sep 18 14:07:56 2018 +0300

    Первый коммит
```

8. Теперь создаем новую ветку.

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git branch testing
```

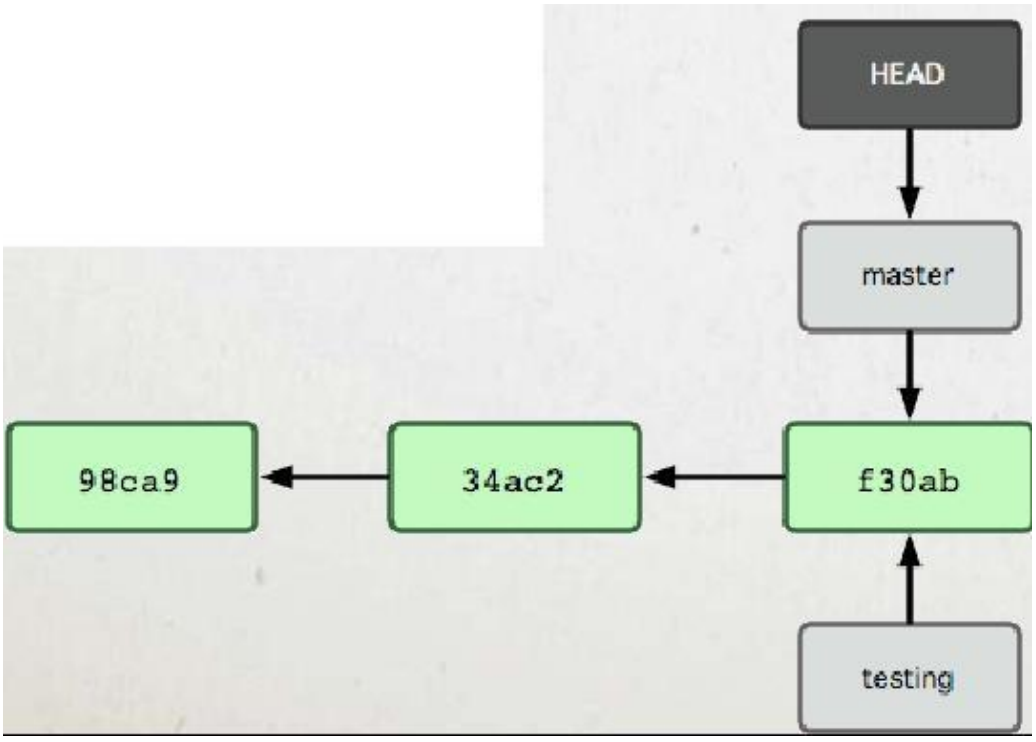
Мы создали новую ветку, которая указывает на тот же коммит, который на скриншоте виден под номером 05b133ad74…

9. Однако, мы до сих пор находимся в ветке master!!!Проверьте это командой git status.Это потому, что команда git branch ветку только создает.

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git status
On branch master
nothing to commit, working tree clean
```

10. Для того, чтобы переключиться на новую ветку, используется команда git checkout. Для её понимания рассмотрим, что такое HEAD.

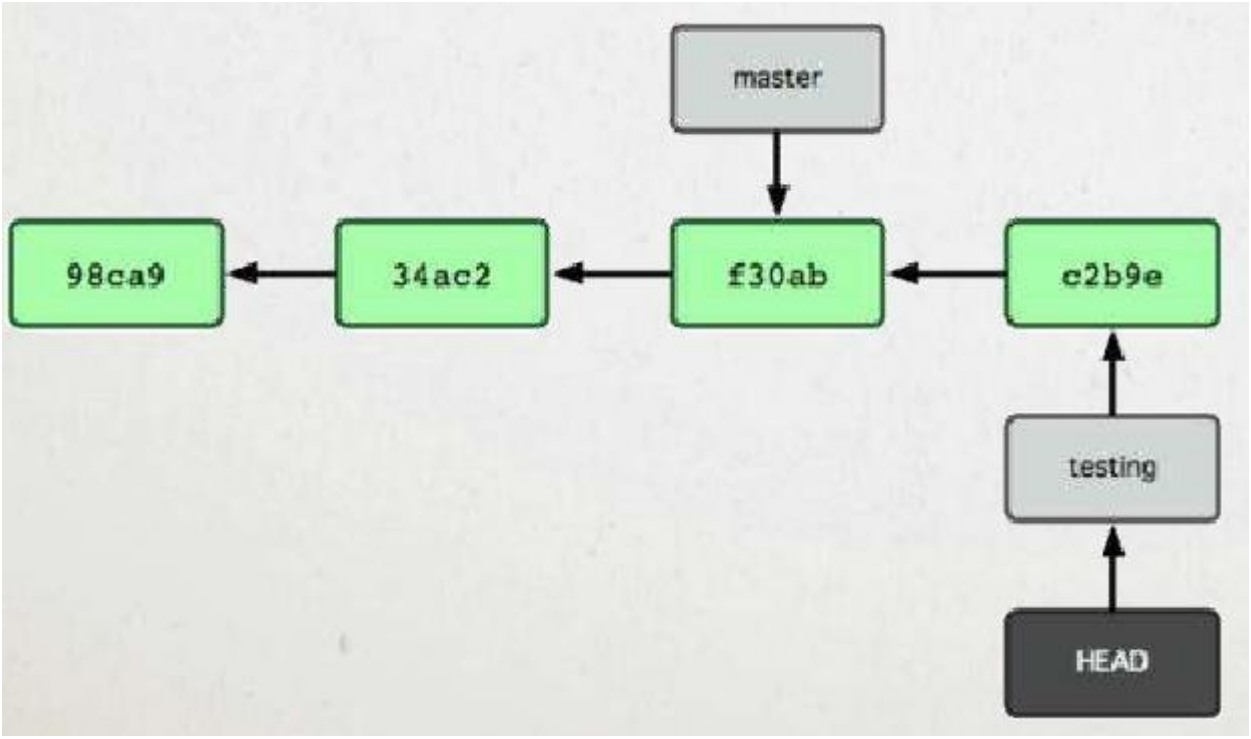
HEAD – это указатель на указатель. Это специальный указатель на текущую ветку и коммит. То есть, это указатель на текущее состояние репозитория (говорят, что это указатель на то, в какой ветке мы находимся. На самом деле мы в ней не находимся, просто текущий указатель указывает на нее). Команда git checkout передвинет HEADна указанную нами ветку.




11. Перейдите на ветку testing.

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git checkout testing
Switched to branch 'testing'
```

В результате предыдущая схема будет выглядеть так:




12. В консоли вы видите текст: «Switched to branch ‘testing’», это означает, что мы переключились на ветку testing.

13. Также можно проверить текущую ветку с помощью gitstatus:

```
Танюшка@NoteBook MINGW64 /d/temp/branches
 git status
On branch testing
nothing to commit, working tree clean
```

Как видите, никаких изменений не произошло, мы просто переключились с одной ветки на другую.

14. Сделаем в ветке testing ещё один коммит. Для этого откройте файл README.txt и допишите в конце любое слово, например, «Тест». Добавьте и индексируйте изменения, сохранив коммит под именем «Коммит в ветке testing».

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git add ./README.txt

Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git commit -m "Коммит в ветке testing"
[testing 450680d] Коммит в ветке testing
 1 file changed, 2 insertions(+), 1 deletion(-)
```

15. Проверьте git status, убедитесь, что все изменения зафиксированы. Выведите git log и вы увидите два коммита.

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git log
commit 450680d12f80565cea4c641d8c49136e7ce3ddd5 (HEAD -> testing)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Sep 18 14:39:30 2018 +0300

    Коммит в ветке testing

commit 05b133ad74ed028634f770e1dd94067aae6d9cbb (master)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Sep 18 14:07:56 2018 +0300

    Первый коммит
```

16. Далее перейдите в ветку master.

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git checkout master
Switched to branch 'master'
```

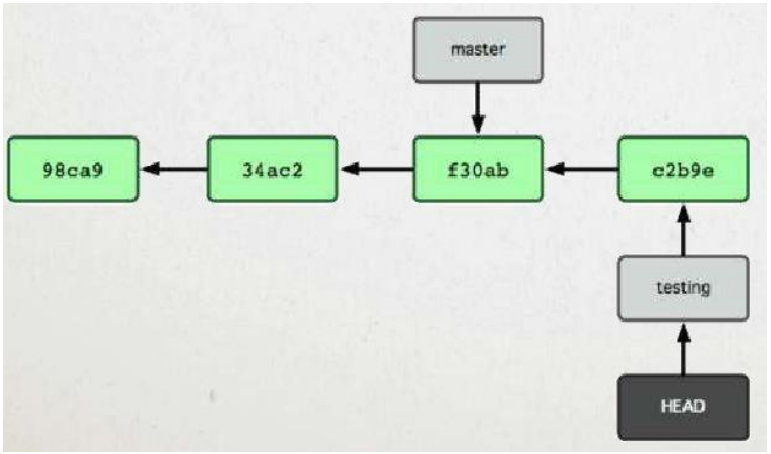
Мы вернули репозиторий в то состояние, с которым была связана ветка master. Чтобы убедиться в этом, откройте файл README.txt, и вы увидите, что дописанный текст «Тест», который был сохранен в ветке testing, исчез. Он существует только в ветке testing.

17. Выведите git log. Вы увидите, что коммит, относящийся к ветке testing, также не отображается.

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git log
commit 05b133ad74ed028634f770e1dd94067aae6d9cbb (HEAD -> master)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Sep 18 14:07:56 2018 +0300

    Первый коммит
```

18. Перейдите опять на ветку testing. Проверьте лог, вы снова увидите два коммита. Откройте файл README.txt, там опять будет две строчки.

Итак, мы находимся в ситуации, представленной на рисунке:



Возникает вопрос, что произойдет, если мы перейдем на ветку master и создадим ещё один коммит? Получится, что у одного из коммитов несколько ветвей:


19. Промоделируем эту ситуацию в нашем репозитории. Перейдите на ветку master и выведите лог.

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git checkout master
Switched to branch 'master'

Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git log
commit 05b133ad74ed028634f770e1dd94067aae6d9cbb (HEAD -> master)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Sep 18 14:07:56 2018 +0300

    Первый коммит
```

20. Внесем изменения в репозиторий. Добавьте файл index.txt, напишите в него текст «Главный файл».

21. Создайте коммит «Коммит в ветке master».

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git add ./index.txt

Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git commit -m "Коммит в ветке master"
[master e915b88] Коммит в ветке master
 1 file changed, 1 insertion(+)
 create mode 100644 index.txt
```

22. Вызовите git log. Вы увидите в ветке master два коммита:

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git log
commit e915b88e7832956d5713b8c762b1e3cdc7c51929 (HEAD -> master)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Fri Sep 21 12:20:54 2018 +0300

    Коммит в ветке master

commit 05b133ad74ed028634f770e1dd94067aae6d9cbb
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Sep 18 14:07:56 2018 +0300

    Первый коммит
```

23. Переключитесь в ветку testing и также посмотрите git log. Вы видите, что второй коммит совершенно другой:

```
Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git checkout testing
Switched to branch 'testing'

Танюшка@NoteBook MINGW64 /d/ProGit/branches
 git log
commit 450680d12f80565cea4c641d8c49136e7ce3ddd5 (HEAD -> testing)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Sep 18 14:39:30 2018 +0300

    Коммит в ветке testing

commit 05b133ad74ed028634f770e1dd94067aae6d9cbb
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Tue Sep 18 14:07:56 2018 +0300

    Первый коммит
```

Таким образом мы получили разветвление в истории.

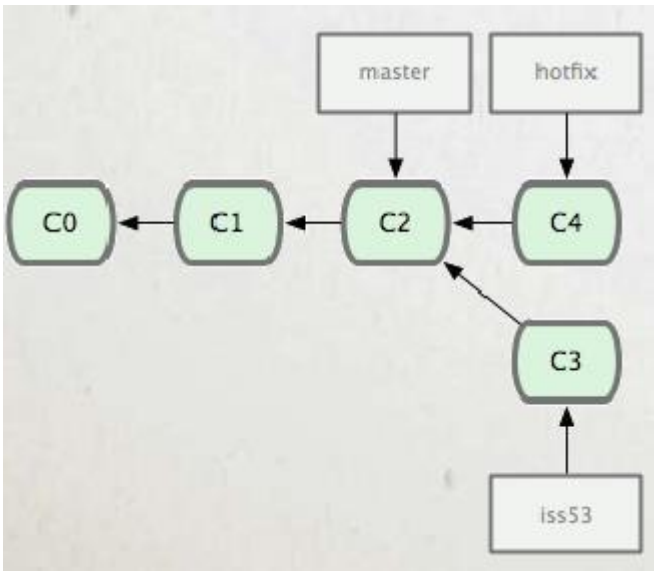
**Важная рекомендация: переключаться между ветками следует в чистом состоянии репозитория! Не оставляйте непроиндексированных изменений! Иначе могут возникнуть проблемы: не произойдет переключения, пропадут незафиксированные данные и др.**

**Однако, если изменения пока не подлежат коммиту, стоит использовать git stash, переключиться, а по возвращении использовать git stash apply. Это стандартный метод работы с Git!!!**

### Слияние веток

Допустим, мы хотим объединить две ветки, слив изменения в одну.

Например, у нас в репозитории ситуация, представленная на рисунке:

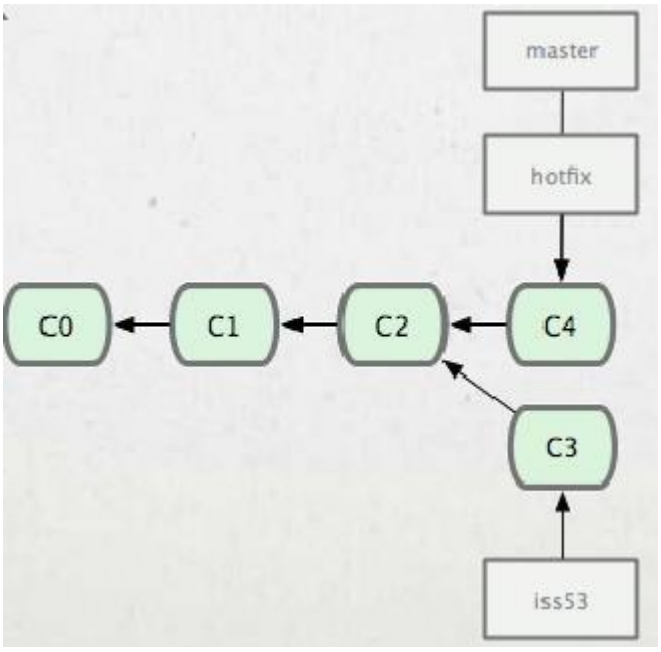



Здесь две ветки – master и hotfix находятся на одной линии истории (iss53 в данный момент не рассматриваем). То есть, в какой-то момент мы сделали ветку hotfix, внесли в нее какие-то изменения, закоммитили их, протестировали, и теперь готовы эти изменения влить в master.

Для этого нужно:

1) переключиться на ветку master (на ту ветку, которая будет принимать изменения);
2) дать команду git merge hotfix (merge – означает «слить»).

После этого Git рассуждает следующим образом: master и hotfix находятся на одной ветке истории. Для слияния их изменений достаточно историю просто перемотать вперёд. Что он и сделает. То есть, в данном случае никакого слияния фактически происходить не будет. Просто указатель мастер тоже переместится на C4:



Такой тип слияния называется fast-forward (ff). Такой сценарий реализуется только тогда, когда у нас ветки находятся на одной линии истории.

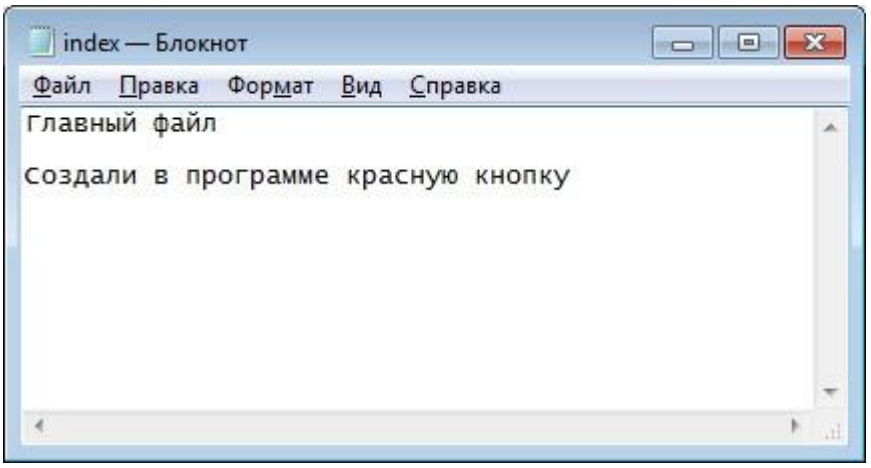
После этого ненужную ветку можно сразу удалить. Для этого используется команда git branch -d *hotfix* (вместо hotfix подставить название своей удаляемой ветки).

Рассмотрим пример.

1. Выведите в ветке master список коммитов, вы увидите 2 коммита.
2. Создадим от этой ветки ветку hotfix (например, нам надо срочно внести какое-то изменение в программу).

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (master)
$ git branch hotfix
```

3. Переключитесь на эту ветку.
4. Добавьте в файле index новый текст:




5. Проверьте git status, вы увидите, что появились изменения.
6. Сделайте коммит «Создали красную кнопку».
7. Проверьте лог, вы увидите все три коммита. При этом третий коммит относится только к ветке hotfix:

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (hotfix)
$ git log
commit c7863fb49abecb7270696496008e8c0d00eae877 (HEAD -> hotfix)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:53:16 2018 +0300

    Создали красную кнопку

commit b2052d1d9887cb4ebacf27c92f56f54603bb9d47 (master)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:44:23 2018 +0300

    Коммит в ветке master

commit 912951b6ccfdf281502c3fafc1bed067619af5f9
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:39:03 2018 +0300

    Первый коммит
```

Допустим, мы внесли таким образом нужные изменения в нашу программу, их в ветке hotfix изучили инженеры по качеству, тестировщики и так далее, после чего, когда эта версия принята, вам как программисту поступает команда влить эти изменения в основную ветку.

8. Переключитесь на ветку master и выведите лог. Вы увидите два коммита ветки master.

9. Выполните слияние:

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (master)
$ git merge hotfix
Updating b2052d1..c7863fb
Fast-forward
 index.txt | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
```

Вы видите, что в 4й строке написано Fast-forward. Это означает, что в результате команды произошла перемотка этой истории, и в её ходе был изменен файл index.txt (5я строка).

10. Проверьте статус. Мы в ветке master, коммиты не требуются.

11. Выведите лог. Вы увидите, что все коммиты появились в ветке master.

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (master)
$ git log
commit c7863fb49abecb7270696496008e8c0d00eae877 (HEAD -> master, hotfix)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:53:16 2018 +0300

    Создали красную кнопку

commit b2052d1d9887cb4ebacf27c92f56f54603bb9d47
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:44:23 2018 +0300

    Коммит в ветке master

commit 912951b6ccfdf281502c3fafc1bed067619af5f9
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:39:03 2018 +0300

    Первый коммит
```

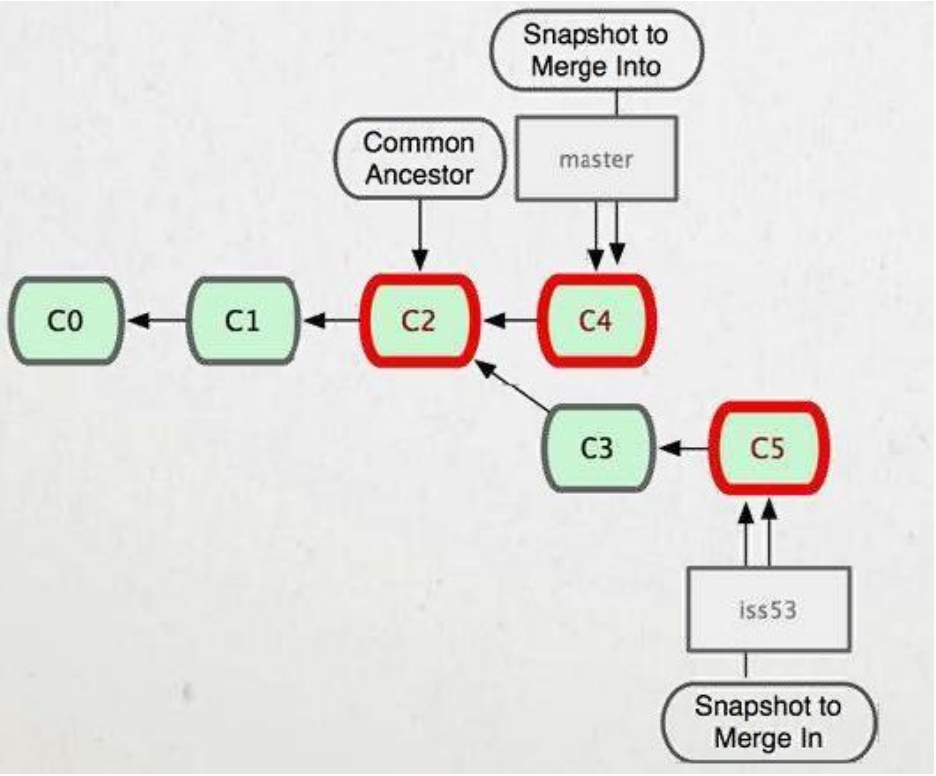
Таким образом, мы рассмотрели самый простой вид слияния – Fast forward.

12. Вызовите команду git branch –v

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (master)
$ git branch -v
  hotfix  c7863fb Создали красную кнопку
* master  c7863fb Создали красную кнопку
  testing 4d6079e Коммит в ветке testing
```

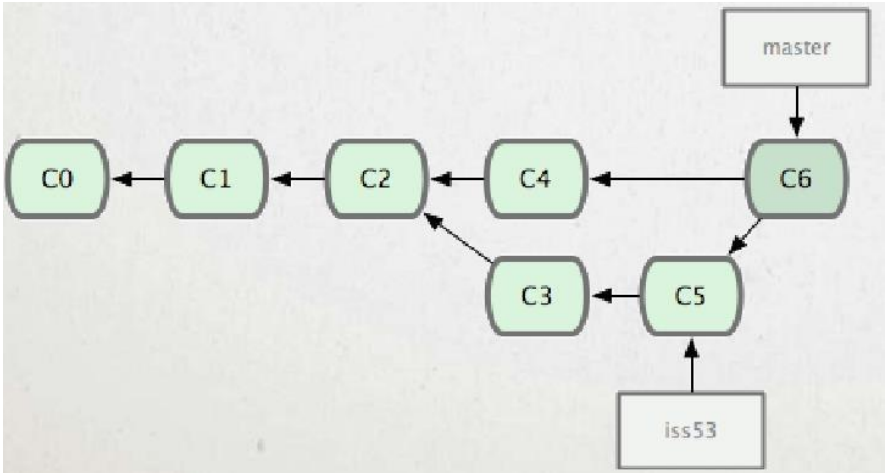
Этак команда выводит полный перечень веток репозитория. Звездочкой отмечена текущая ветка, на которую смотрит HEAD.

Теперь рассмотрим более сложный случай: как сливать изменения, если ветки уже разошлись? Допустим, что ветка master уже ушла вперёд от общего родителя (C4). А вторая ветка Iss53 уже ушла от неё на два коммита (C5). То есть, эти ветки не находятся на одной линии истории.




В этом случае git применяет алгоритм «Наилучшего общего предка». От обеих веток git просматривает историю назад, пока не найдет ближайшего общего предка. После этого он начинает последовательно к этому предку применять коммиты, ориентируясь на время, когда они были созданы, и таким образом создает новый коммит слияния.

Таким образом, в отличие от Fast forward, когда ничего нового не создавалось, здесь создастся новое изменение. Создастся новый коммит C6 в ветке master, который соберет в себе все изменения C3, C4 и C5:




Рассмотрим практический пример.

13. Проверьте статус, вы должны находиться в ветке master:

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (master)
$ git status
On branch master
nothing to commit, working tree clean
```

14. Она уже ушла достаточно далеко от ветки testing, созданной ранее. Убедимся в этом, выведем лог для ветки master, затем перейдем в ветку testing и проверим лог для неё.

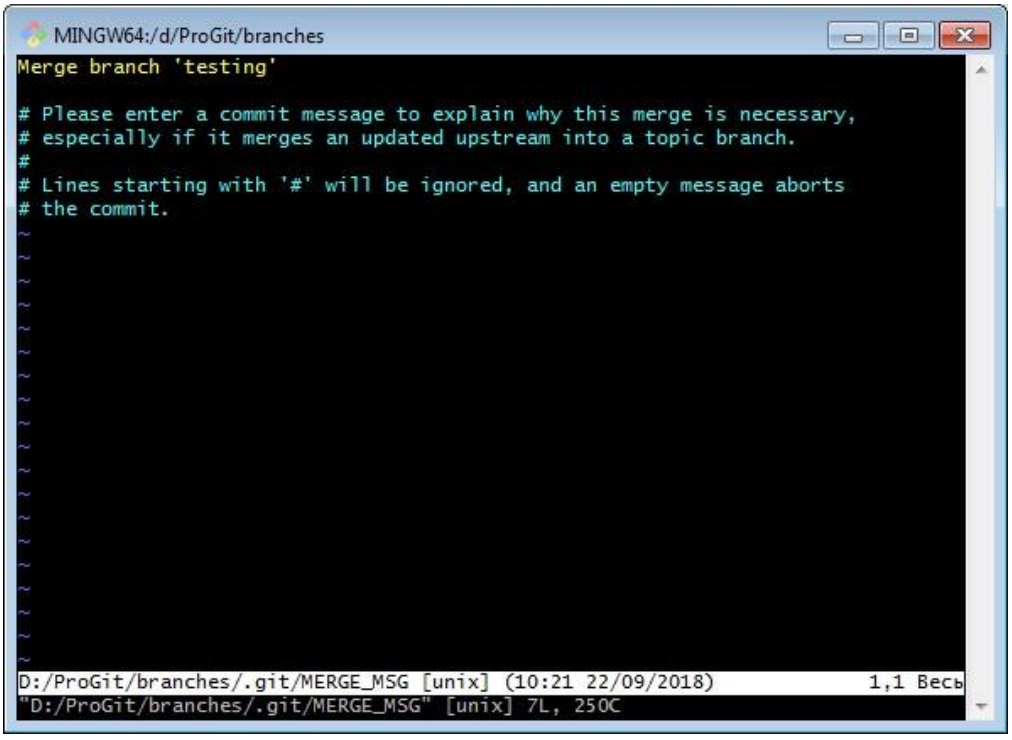
```
d103@S-T000012680 MINGW64 /d/ProGit/branches (master)
$ git log
commit c7863fb49abecb7270696496008e8c0d00eae877 (HEAD -> master, hotfix)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:53:16 2018 +0300

    Создали красную кнопку

commit b2052d1d9887cb4ebacf27c92f56f54603bb9d47
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:44:23 2018 +0300

    Коммит в ветке master

commit 912951b6ccfdf281502c3fafc1bed067619af5f9
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:39:03 2018 +0300

    Первый коммит
```

Переходим в testing:

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (master)
$ git checkout testing
Switched to branch 'testing'

d103@S-T000012680 MINGW64 /d/ProGit/branches (testing)
$ git log
commit 4d6079ec915327ceb8f4f719587e0d0ffede2ebe (HEAD -> testing)
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:41:53 2018 +0300

    Коммит в ветке testing

commit 912951b6ccfdf281502c3fafc1bed067619af5f9
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 22 08:39:03 2018 +0300

    Первый коммит
```

У них общий только первый коммит, в остальном они разошлись, у каждого своя история.

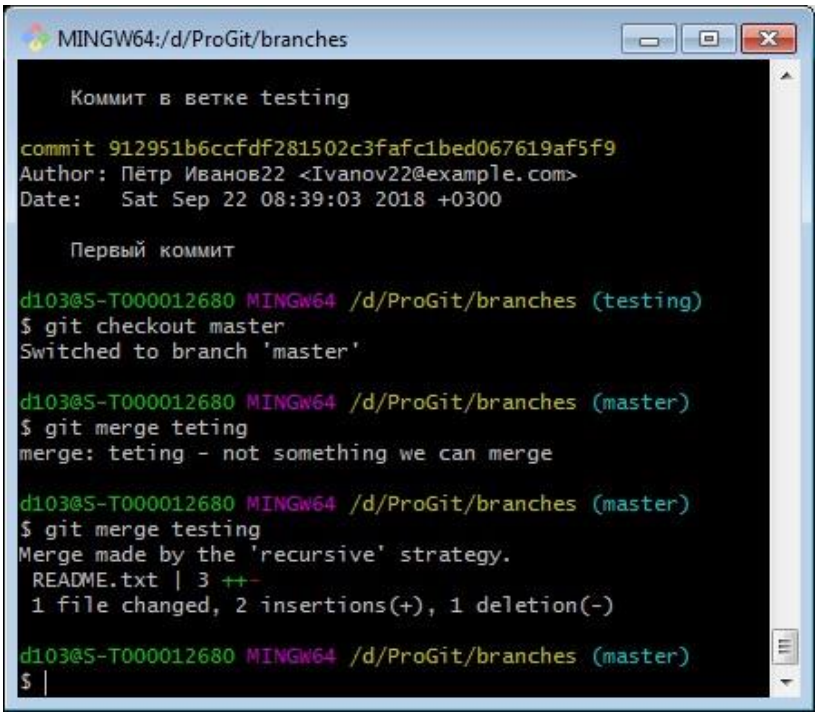
15. Снова переключаемся в master и вливаем неё ветку testing (git merge testing). Начнется слияние:




16. Слияние удалось и нам открылся текстовый редактор, который просит нас ввести сообщение для коммита. Чтобы выйти из него, нажмите последовательно <esc> <:> <q> (если в качестве текстового редактора по умолчанию у вас установлен другой, то вы можете просто закрыть его и Git Bash вернется в рабочий режим).

Также, если надо записать изменения в этом текстовом редакторе, нужно нажать <esc> <:> <w> (сейчас этого не делайте).

17. Вернулись в Git Bash




Слияние было осуществлено с использованием рекурсивной стратегии.

18. Выведите лог. Вы видите перечень того, какие коммиты применялись, последним создан коммит «Merge branch ‘testing’». Это название и можно было поменять в текстовом редакторе, но мы не стали этого делать.

19. Нажмите q для выхода в Git Bash.

Таким образом, Git при слиянии создает новый коммит и этим коммитом отмечает момент, когда это слияние было произведено. Это удобно для того, чтобы в случае необходимости это слияние было легко отменено, и вы могли сдвинуться на момент до него.

Итак, **в Git создание и слияние веток – это основная штатная операция. Если вы хотите внести какие-то изменения в проект, следует создать ветку, внести в ней изменения, протестировать, а затем, если всё нормально – влить в основную, если нет, удалить. Это стандартный метод работы с Git.**

**Главный принцип: одна задача – это одна ветка!**

### Решение конфликтов

Конфликт возникает, если в одном и том же месте (файле) есть разные изменения. Это также штатная ситуация, Git предназначен для разрешения конфликтов.

Конфликт означает, что Git не смог слить версии автоматически и вам необходимо это сделать вручную.

Рассмотрим, что делать в случае возникновения конфликта.

1. Создадим конфликт искусственно. Проверьте, что вы находитесь в ветке master.
2. Создайте в папке branches текстовый файл first.txt и напишите внутри «Это первая строчка.».
3. Сделайте коммит.

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (master)
$ git add first.txt

d103@S-T000012680 MINGW64 /d/ProGit/branches (master)
$ git commit -m "Создали файл"
[master d081c15] Создали файл
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 first.txt
```

4. Сделайте ветку «feature/1» (Задача 1) и переключитесь на нее:

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (master)
$ git checkout -b feature/1
Switched to a new branch 'feature/1'
```

5. Пусть наша задача №1 – сделать красную кнопку. Откройте файл first.txt и напишите в конце «Красная кнопка». Сохраните файл.

Проиндексируйте и закоммитьте изменения в текущей ветке feature/1.

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (feature/1)
$ git add first.txt

d103@S-T000012680 MINGW64 /d/ProGit/branches (feature/1)
$ git commit -m "Создали красную кнопку"
[feature/1 d71c1cc] Создали красную кнопку
 1 file changed, 2 insertions(+)
```

6. Снова переключитесь на ветку master.
7. Создайте ветку под названием feature/2 и перейдите на неё.
8. Вторая задача – сделать две зелёных кнопки. Откройте файл first.txt, там в данной ветке должна быть только одна строка. Напишите во второй строке «Тут две зелёных кнопки».
9. Создайте коммит:

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (feature/2)
$ git add first.txt

d103@S-T000012680 MINGW64 /d/ProGit/branches (feature/2)
$ git commit -m "Создали две зеленых кнопки"
[feature/2 f3fc0e9] Создали две зеленых кнопки
 1 file changed, 2 insertions(+)
```

Итак, у нас есть две ветки, у которых разошлась история.

10. Переключитесь на первую задачу:

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (feature/2)
$ git checkout feature/1
Switched to branch 'feature/1'
```

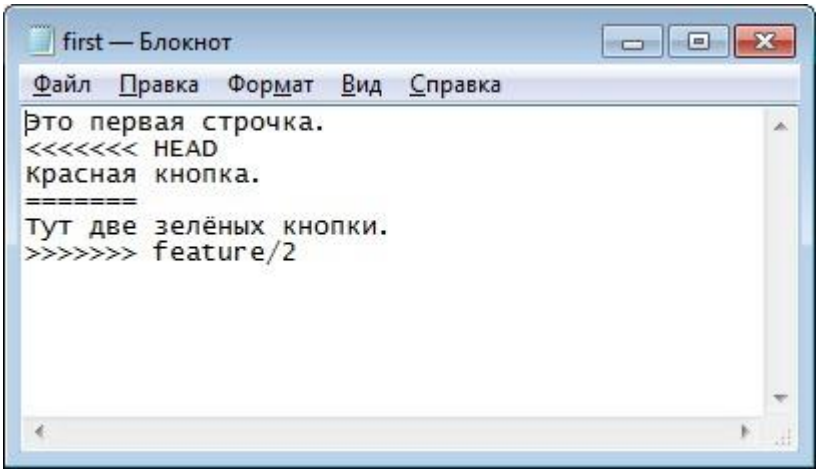
11. Попробуем в эту ветку влить изменения из второй.

```
d103@S-T000012680 MINGW64 /d/ProGit/branches (feature/1)
$ git merge feature/2
Auto-merging first.txt
CONFLICT (content): Merge conflict in first.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Мы видим надпись CONFLICT – в файле first.txt возник mergeконфликт. Автоматическое слияние невозможно. Нужно поправить конфликты и закоммитить результат.

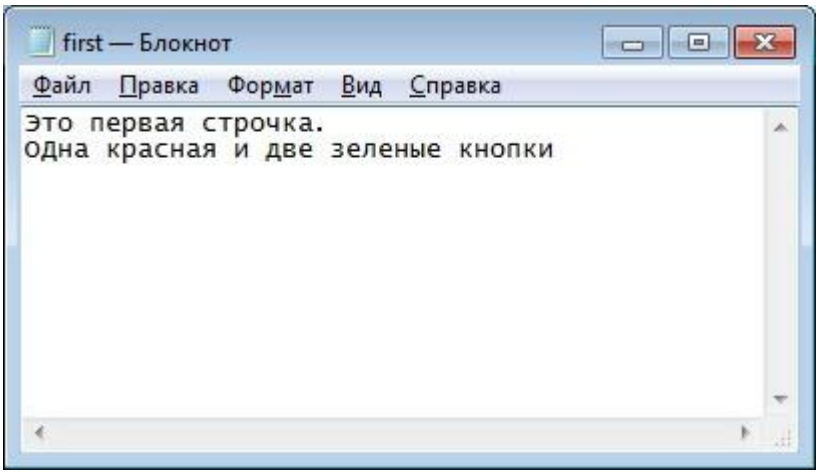
То есть, Git попытался применить последовательную стратегию слияния, обнаружил противоречия и вывел такое сообщение.

12. Откройте файл first.txt и вы увидите следующую картину:



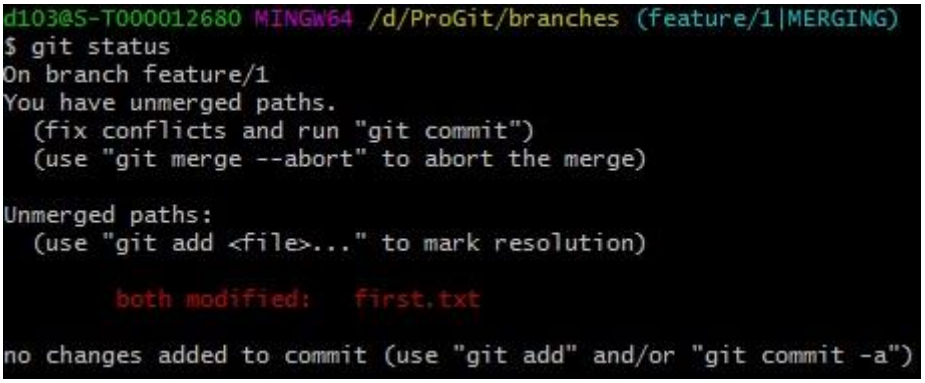

13. В файле появились метки, которые указывают, что же вызвало конфликт, что именно вызывает противоречие, а именно: Красная кнопка  Две зеленые кнопки.

14. Git хочет, чтобы мы оставили что-то одно, требует выбрать некий результат вручную (или оба). Сотрите всю служебную секцию и напишите «Одна красная и две зеленые кнопки»:



Закройте и сохраните файл.

15. Проверьте статус репозитория.




16. Проиндексируйте файл и закоммитьте его под именем «Конфликт решён».

ДОМАШНЯЯ РАБОТА

1. Начните новый репозиторий на Гитхабе

2. Пользуясь своими знаниями HTML и CSS, создайте в нем файл index.html Пусть в этом файле будет информация о вас, профессиональный профиль. (достаточно пары абзацев, суть ДЗ не в том, чтобы красиво сверстать страницу). Зафиксируйте изменения.

3. Представьте теперь, что руководство ставит задачу: разместить на странице кнопку "Подписаться на новости"

- Создайте новую ветку для выполнения этой задачи
- Добавьте кнопку. Зафиксируйте изменения. Передайте их на гитхаб и убедитесь, что вы видите новую ветку в интерфейсе Гитхаба.
- Влейте изменения из ветки задачи в ветку master - вы должны увидеть merge методом fast-forward
- Передайте все изменения в удаленный репозиторий

4. Поступили сразу две новых задачи: изменить цвет кнопки из пункта 3. и добавить на страницу форму входа через социальные сети (условно, конечно!)

- Создайте для каждой задачи свои ветки
- Выполните их
- Влейте изменения в master
- Передайте изменения в Github

5. Смоделируйте возникновение конфликта, внося изменения в одно и то же место своего файла. Решите конфликт.

*Контрольные вопросы:*

1. Какая ветка создается сразу после создания репозитория?
2. Как задать новую ветку?
3. Что такое head и какая взаимосвязь между git checkout?
4. Как произвести слияние веток?
5. Что значит fast-forward?

*Список литературы:*


1. Гохберг, Г. С. Информационные технологии : учебник / Г.С. Гохберг, А.В. Зафиевский, А.А. Короткин. - 9-е изд., перераб. и доп. - М. : Академия, 2014. - 240 с.
2. Белов, В. В. Проектирование информационных систем : учебник / В.В. Белов, В.И. Чистяков ; под ред. В.В. Белова. - М. : Академия, 2013. -
352 с. - (Бакалавриат). - На учебнике гриф: Рек.УМО. - Библиогр.: с.
345-347. - ISBN 978-5-7695-7406-1.


**Практическая работа №7. Работа с GitHub.**

*Цель работы:*

Регистрация на Github. Создание репозитория, клонирование. Команды Github’a.

*Теоретическая часть.*

В первую очередь надо установить клиент git: обязательно потребуется консольный клиент, доступный по ссылке http://git-scm.com/downloads
(поддерживаются основные ОС), графический клиент можно установить по желанию, исходя из своих предпочтений. На Unix системах можно воспользоваться менеджером пакетов (yum на fedora и подобных или apt-get на debian, ubuntu и подобных) вместо того, чтобы скачивать установщик с сайта.

Далее работа с git будет объясняться на примере работы с консольным клиентом по следующим причинам:

- Чтобы у вас складывалось понимание происходящего и при возникновении проблем вы могли четко объяснить, что вы делали, и было видно, что пошло не так.
- Все нажатия кнопок в графических клиентах в итоге сводят к выполнению определённых команд консольного клиента, в то же время возможности графических клиентов ограничены по сравнению с консольным
- У тех, кто будет работать в классе на стоящих там компьютерах, не будет другого выбора, кроме как пользоваться консольным клиентом (на сколько мне известно, никаких графических клиентов для git там не установлено)

*Практическая часть.*

В данной работе рассмотрим использование для этих целей популярного в настоящее время сервиса Github.

GitHub — крупнейший веб-сервис для хостинга IT-проектов и их совместной разработки. Веб-сервис основан на системе контроля версий Git и разработан на RubyonRails и Erlang компанией GitHub, Inc. Википедия

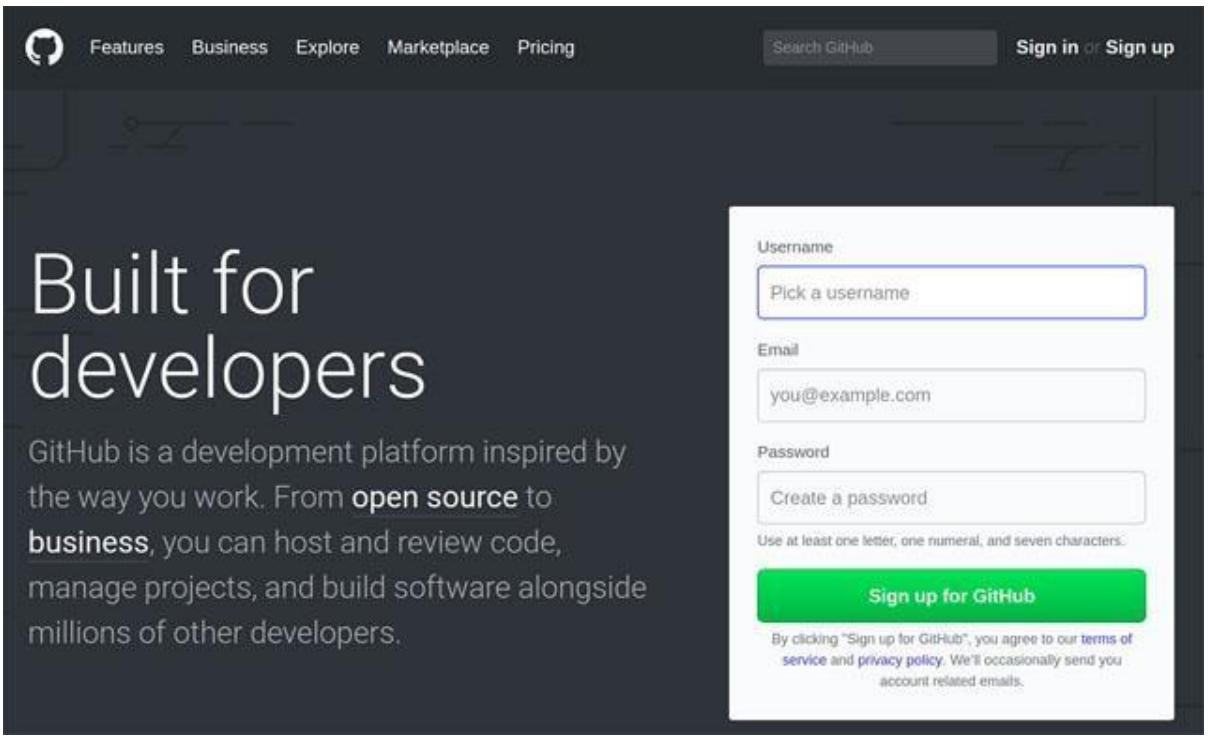
На его примере разберем, как работать с получением изменений и их отправкой.

**Регистрация на GitHub**

**GitHub** — веб-сервис, который основан на системе Git. Это такая социальная сеть для разработчиков, которая помогает удобно вести коллективную разработку IT-проектов. Здесь можно публиковать и редактировать свой код, комментировать чужие наработки, следить за новостями других пользователей. Именно в GitHub работаем мы, команда Академии, и студенты интенсивов.

Чтобы начать работу с GitHub, нужно зарегистрироваться на сайте, если вы ещё этого не сделали. За дело.

1. Переходим на сайт GitHub.

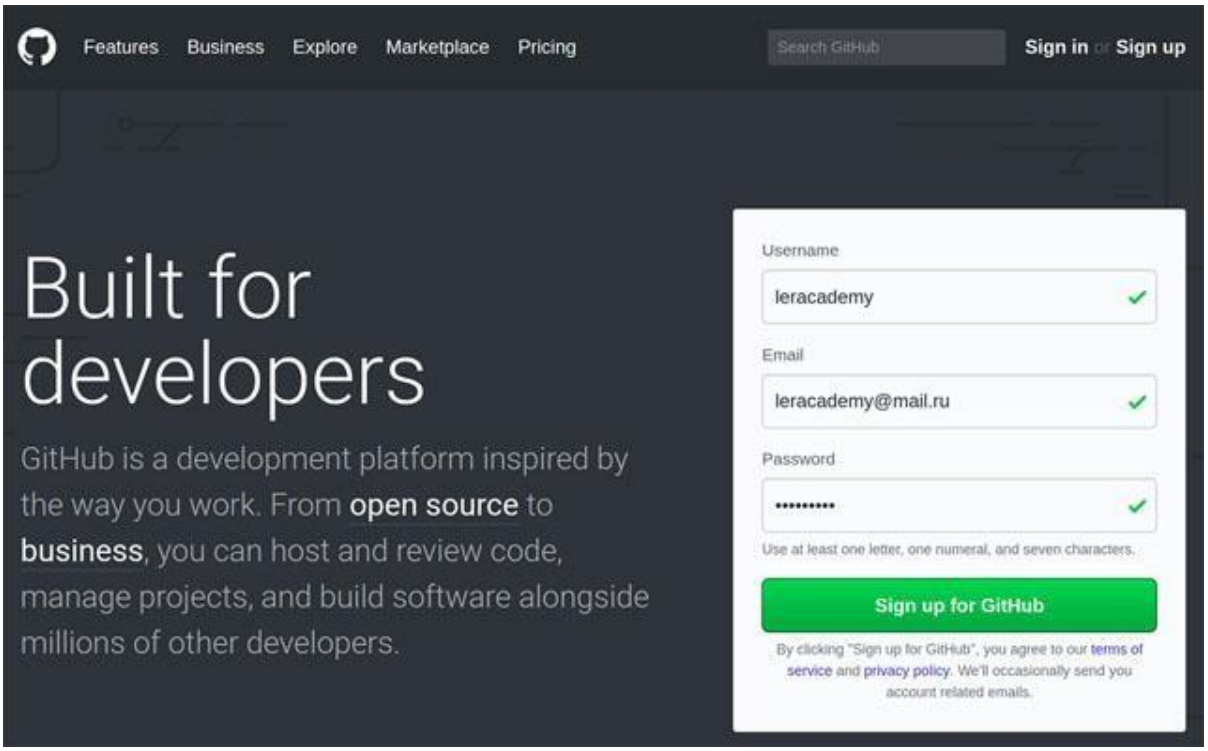



Стартовая страница GitHub.

2. Есть два варианта начала регистрации:

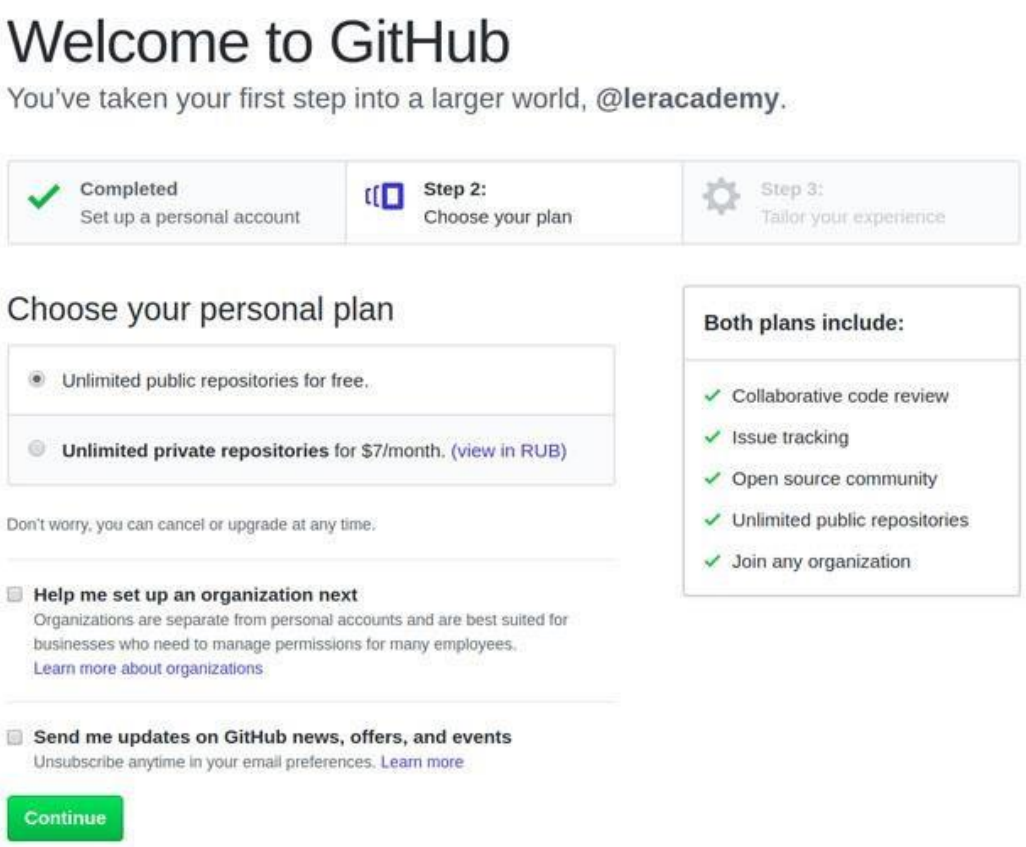
- Нажимаем кнопку Sign up (зарегистрироваться), попадаем на страницу регистрации, где вводим обязательные данные: имя пользователя, адрес электронной почты и пароль. После заполнения полей нажимаем Createan account (создать аккаунт).
- Сразу вводим имя, почту и пароль на главной странице GitHub и

нажимаем Signup forGitHub (зарегистрироваться на GitHub).




Первый шаг регистрации профиля на стартовой странице GitHub.

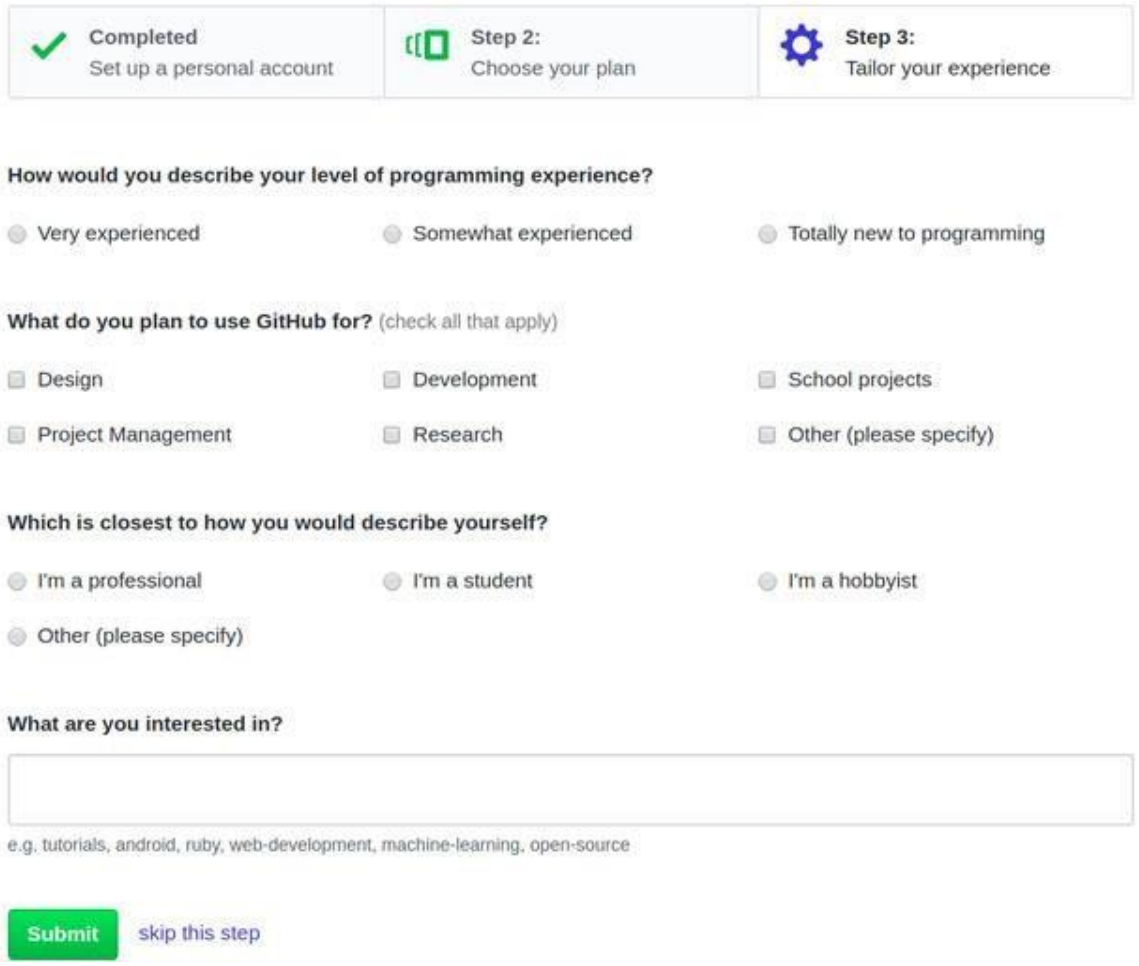
3. На втором этапе нужно выбрать тарифный план. GitHub — бесплатный сервис, но предоставляет некоторые платные возможности. Выбираем тарифный план и продолжаем регистрацию.



Выбор тарифа на втором шаге регистрации.

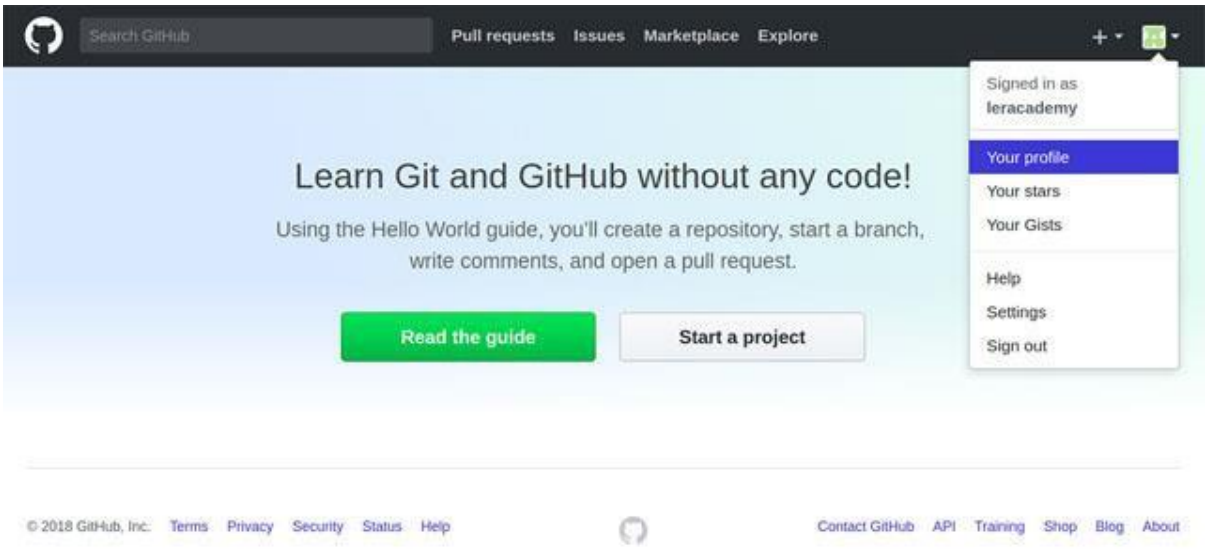
Пользование GitHub бесплатно до тех пор, пока ваш код открыт для всех. Платные версии позволяют создавать закрытые, приватные репозитории, которые видите только вы.

4. Третий шаг — небольшой опрос от GitHub, который вы можете пройти, заполнив все поля и нажать Submit или пропустить, нажав skipthisstep.

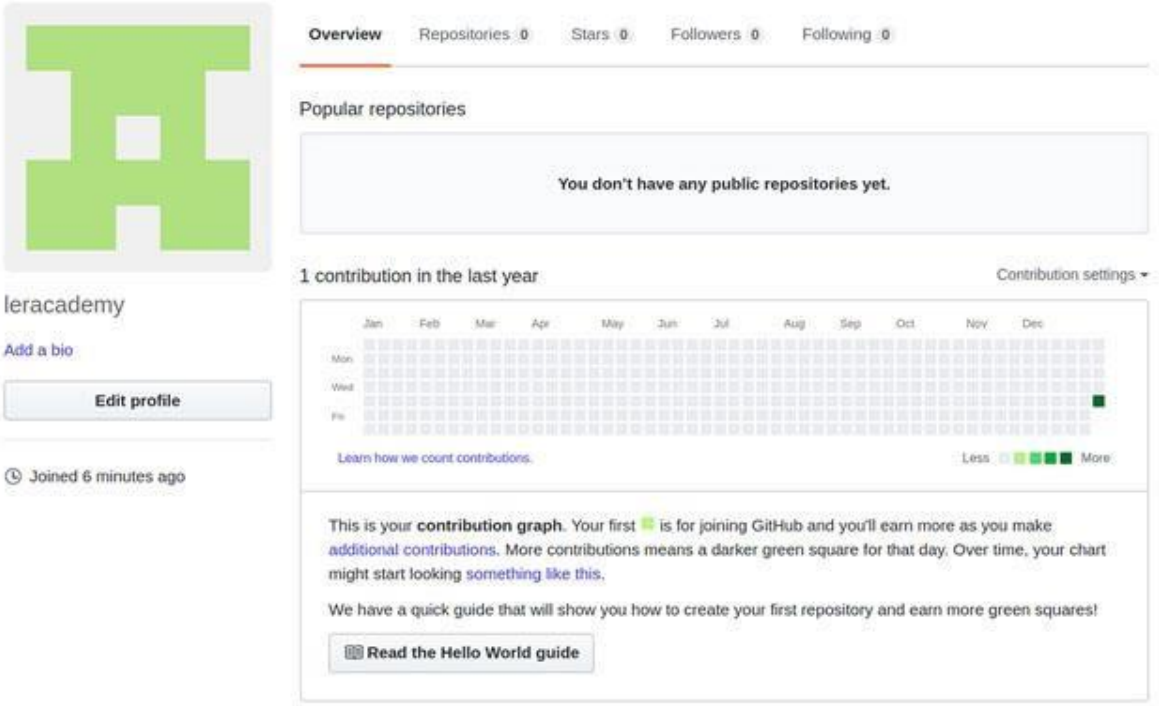



5. Опрос на третьем шаге регистрации.

6. После прохождения всех этапов на сайте, на указанный при регистрации ящик вам придёт письмо от GitHub. Откройте его и подтвердите свой почтовый адрес, нажав Verifyemailaddress (подтвердить электронный адрес) или скопируйте вспомогательную ссылку из письма и вставьте её в адресную строку браузера.




Переход в ваш профиль.




Так выглядит ваш профиль после регистрации.

Теперь у вас есть профиль на GitHub.

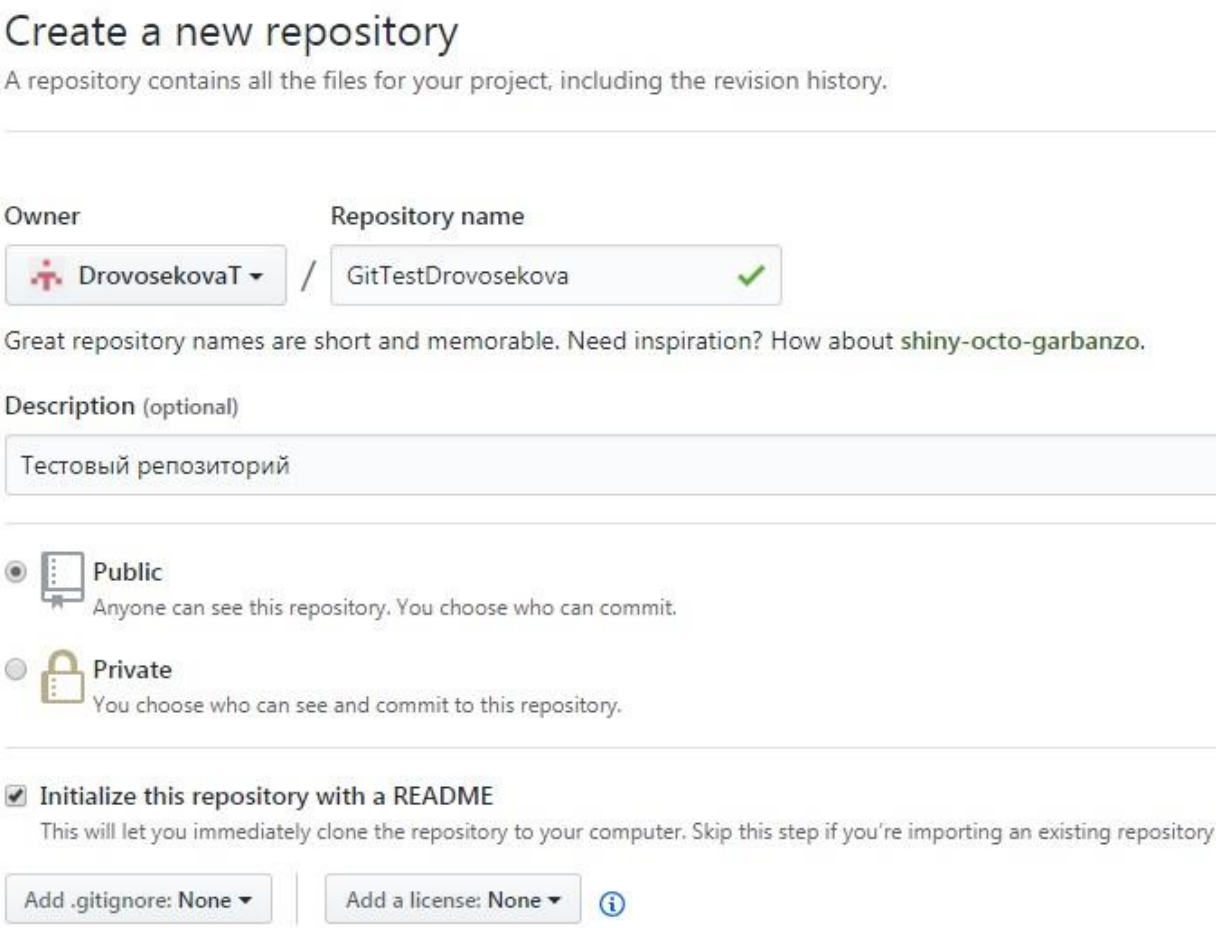
Сейчас у нас нет ни одного репозитория, и мы можем либо создать новый репозиторий, либо ответвиться (fork) от уже существующего чужого репозитория и вести собственную ветку разработки. Затем, при желании, свои изменения можно предложить автору исходного репозитория
(Pullrequest).

7. Если Вы находитесь в своем профиле, то в верхней части окна видите один из пунктов меню «Repositories 0». Это означает, что у вас в данный момент нет созданных репозиториев. На этой же

вкладке есть кнопка «New», которая позволяет создать новый репозиторий. Нажимаем её.

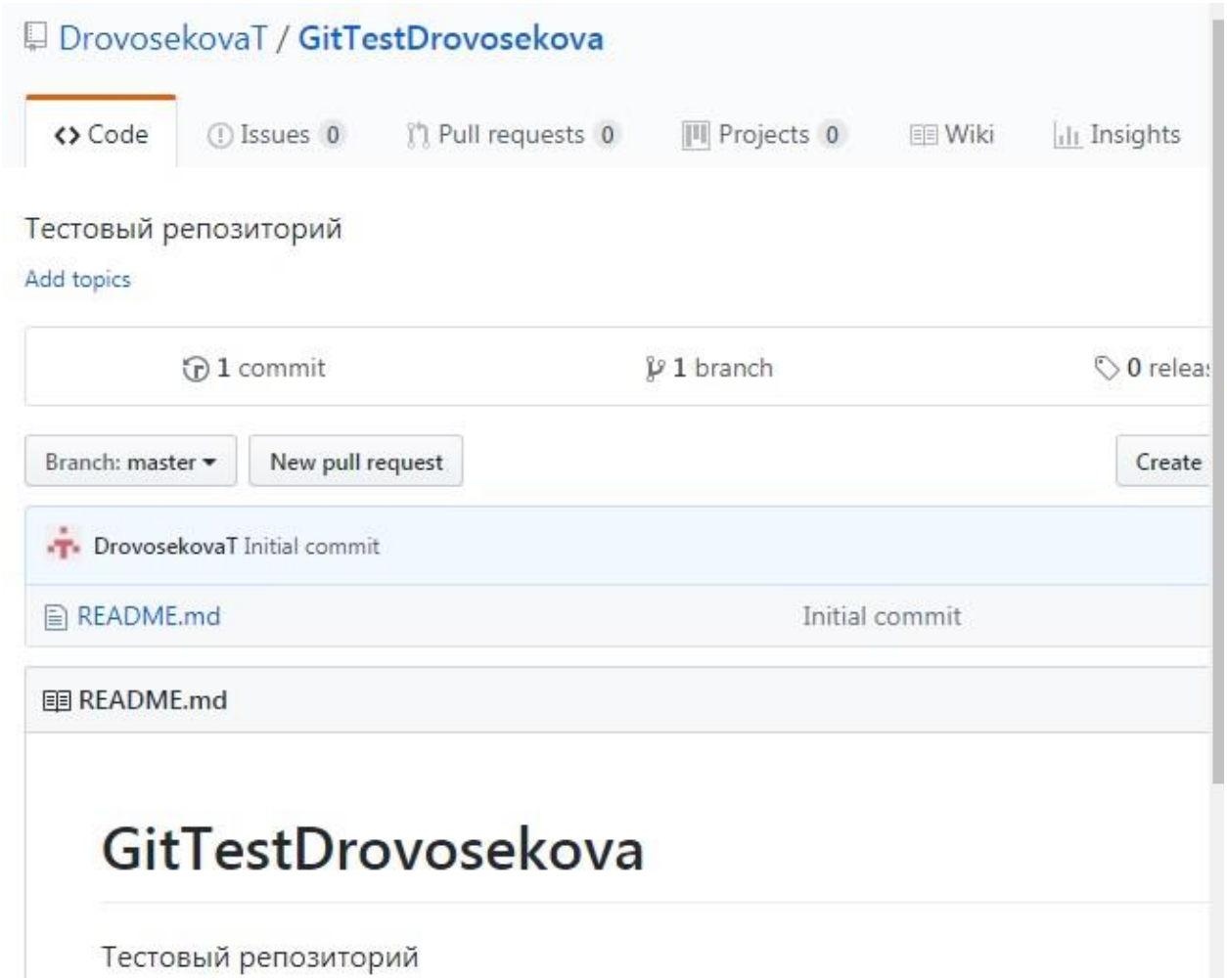
8. Открылось окно создания репозитория. Даем ему имя (например, gittest) можно написать описание. Он должен быть публичным. Также установите галочку там, где предлагается поместить в репозиторий текстовый файл Readme.

Нажмите кнопку «Create repository».

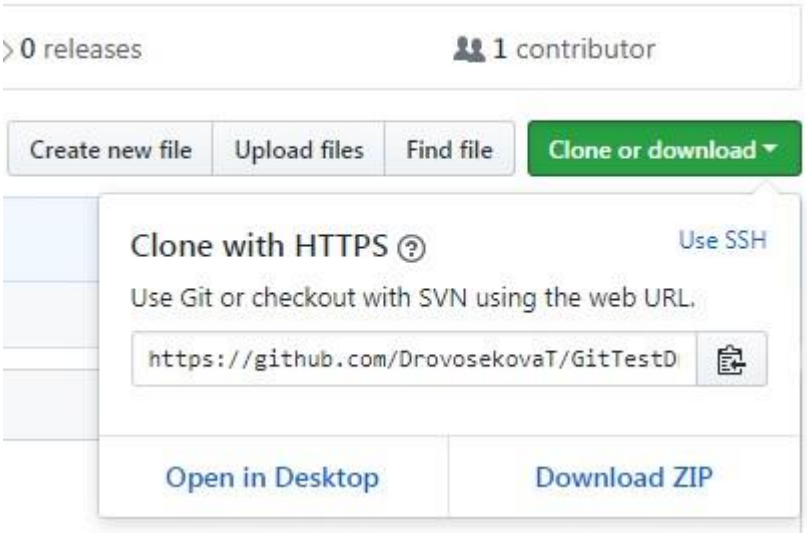



9. Когда вы создаете новый репозиторий таким образом, на Github появляется папка и для нее выполняется команда gitinit, т.е. происходит то же самое, что мы делали в предыдущих лабораторных работах вручную.

10. После создания репозитория вы в него попадаете и видите, что в нем в данный момент находится один файл – README.md, который был создан автоматически:




11. Далее нам нужно скопировать в буфер обмена адрес нашего репозитория на Github. Для этого нажмите кнопку «Cloneordownload»и в открывшемся окне копку справа от адреса ссылки:




12. Откройте консоль GitBash (она должна быть у вас открыта после лабораторной работы 2, если нет – перейдите в папку tmp, с

которой мы работали). Даем команду **git clone**https://github.com/DrovosekovaT/GitTestDrovosekova.git **test**

(вставьте из буфера обмена свой адрес репозитория и после него через пробел название папки, где будет создан клон удаленной папки-репозитория, в данном случае test).

Эта команда создаст в папке test копию репозитория с указанного адреса с Github, автоматически Github подключит к локальному репозиторию как удалённый и назовёт его origin, т.е. оригинал (перейдите в папку test и проверьте это командой gitremote -v), автоматически в нашем локальном репозитории создаст локальную ветку master (проверьте это командой gitstatus), которую свяжет с веткой master в удаленном репозитории на Github.

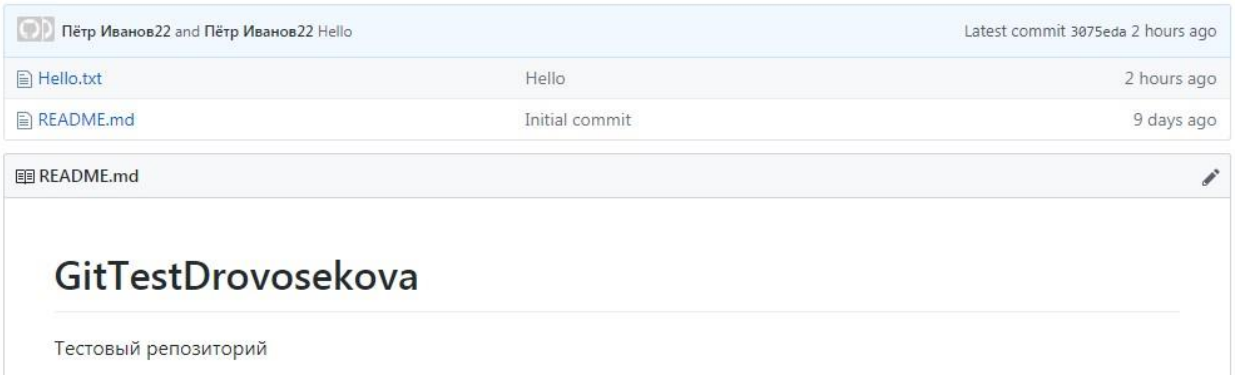
13. Теперь рассмотрим, как синхронизировать локальный репозиторий (клон) с оригиналом, который находится на Github. В клонированной папке test создайте локально файл Hello.txt, напишите в нем текст «Hello!!!», закройте его, сохранив изменения. Теперь у нас репозиторий содержит 2 файла, а на Github только один. Для того, чтобы зафиксировать изменения, выполните команду gitaddHello.txtи создайте коммит:

```
Танюшка@NoteBook MINGW64 /d/tmp/test
 git add Hello.txt

Танюшка@NoteBook MINGW64 /d/tmp/test
 git commit -m "Hello"
[master 3075eda] Hello
 1 file changed, 1 insertion(+)
 create mode 100644 Hello.txt
```

14. Выполните команду **gitpush**. Эта команда берет имеющиеся локальные изменения и отправляет их в удалённый репозиторий. В ответ на запрос введите своё имя на Github и пароль.

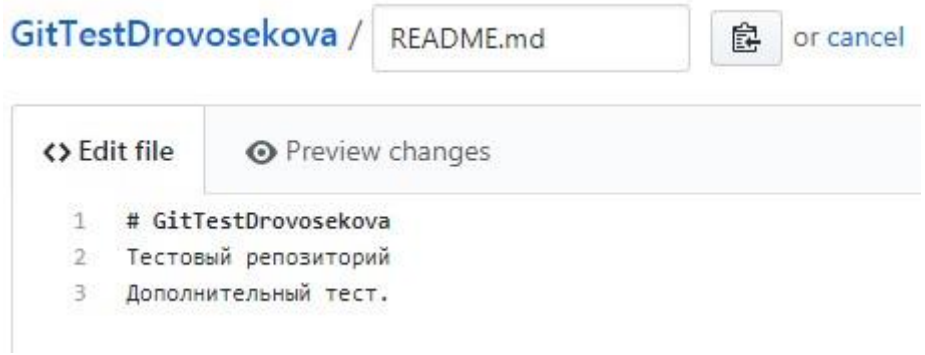
```
Танюшка@NoteBook MINGW64 /d/tmp/test
 git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 306 bytes | 306.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/DrovosekovaT/GitTestDrovosekova.git
   2519672..3075eda  master -> master

Танюшка@NoteBook MINGW64 /d/tmp/test
```

15. Откройте репозиторий на сайте Github, убедитесь, что файл добавился:



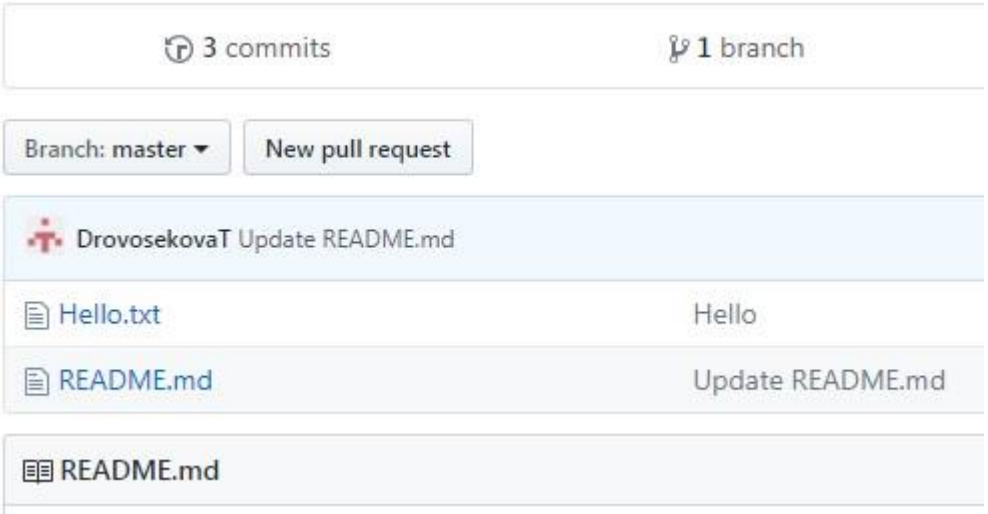

Если вы откроете добавившийся файл Hello.txt, то увидите свой текст. Также вы видите название коммита во втором столбце.

16. Также есть обратная команда. Допустим, у нас на Github появились какие-то изменения, которых нет на локальном компьютере. Для примера откройте и отредактируйте файл README.md. Для этого откройте его на Github и нажмите кнопку с карандашом (Editthisfile), допишите в третьей строке какой-либо текст:




Теперь нажмите кнопку «Commitchanges» в нижней части страницы.

17. Перейдите в папку репозитория, вы увидите, что коммитов стало три.



18. Теперь откройте GitBash на вашем локальном компьютере и вызовите команду gitlog. Вы увидите, что коммитов по-прежнему два. Необходимо загрузить изменения с Github.

19. Введите команду gitpull. Эта команда загрузит изменения из удалённого репозитория.

```
Танюшка@NoteBook MINGW64 /d/tmp/test
 git log
commit 3075eda5bb3f100642e5ed6e3bd2c7f01419b7d0 (HEAD -> master, origin/ma
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 1 13:29:15 2018 +0300

    Hello

commit 2519672015349dcc5622c473733df2f40365ba66
Author: DrovosekovaT <42639140+DrovosekovaT@users.noreply.github.com>
Date:   Thu Aug 23 13:36:27 2018 +0300

    Initial commit

Танюшка@NoteBook MINGW64 /d/tmp/test
 git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/DrovosekovaT/GitTestDrovosekova
   3075eda..ca53ead  master     -> origin/master
Updating 3075eda..ca53ead
Fast-forward
 README.md | 1 +
 1 file changed, 1 insertion(+)
```

20. Вызовите gitlog и убедитесь, что коммитов стало три.

```
Танюшка@NoteBook MINGW64 /d/tmp/test
 git log
commit ca53ead5d8d08764a7e3b5e8dd5ea30e1fd77fef (HEAD -> master, origi
Author: DrovosekovaT <42639140+DrovosekovaT@users.noreply.github.com>
Date:   Sat Sep 1 15:20:31 2018 +0300

    Update README.md

commit 3075eda5bb3f100642e5ed6e3bd2c7f01419b7d0
Author: Пётр Иванов22 <Ivanov22@example.com>
Date:   Sat Sep 1 13:29:15 2018 +0300

    Hello

commit 2519672015349dcc5622c473733df2f40365ba66
Author: DrovosekovaT <42639140+DrovosekovaT@users.noreply.github.com>
Date:   Thu Aug 23 13:36:27 2018 +0300

    Initial commit
```

21. Чтобы посмотреть содержимое обновленного файла, введите команду catREADME.md:

```
Танюшка@NoteBook MINGW64 /d/tmp/test
 cat README.md
# GitTestDrovosekova
Тестовый репозиторий
Дополнительный тест.
```

Примечание: Если у вас уже есть репозиторий на компьютере, то вы можетеего клонировать в пустую (без файла README) папку

на Github. Итоги:

gitpull – не просто «достает изменения» из удаленного репозитория, но и пытается применить изменения из удалённого репозитория к нашему (влить удалённую ветку в локальную). Ветки будут изучаться далее.

gitpush – отправляет наши наборы изменений на удалёный репозиторий.

Таким образом, для чего применяется Github в реальной работе?

1. Githubочень удобно использовать для быстрого бесплатного создания репозитория в интернете.

Для чего может понадобиться репозиторий в интернете?

1. Дать кому-то на него ссылку, чтобы этот человек мог удобно посмотреть на код, хранящийся там, удобно посмотрел на историю изменений.
2. Githubможет служить посредником для коллективной работы.
3. Если вы работаете за несколькими компьютерами, можно клонировать на них репозиторий с Github и синхронизировать через него все версии.

САМОСТОЯТЕЛЬНАЯ РАБОТА

Git, как распределенная система

Мы узнали, что такое Git. Но почему все-таки это распределенная или, как еще говорят, децентрализованная система? Узнаем на уроке!

- понятие удаленного репозитория
- настройка связи между репозиториями
- команды push и pull
- команда fetch

Кроме того мы научимся пользоваться сервисом GitHub и создадим на нем учебный проект

ДОМАШНЯЯ РАБОТА

1. Создайте у себя на компьютере еще один репозиторий ("второй"). Пустой. Свяжите его с уже существующим ("первым") с

помощью git remote. Передайте в него изменения из первого. Объясните полученный результат.

2. Создайте на гитхабе пустой репозиторий ("А"). ВНИМАНИЕ! Не создавайте в нем файл readme!

- Установите его в качестве удаленного для репозитория "первый"
- Передайте из "первого" в "А" изменения
- Приложите ссылку на "А" к своему ДЗ

3. Создайте на гитхабе новый непустой (с файлом readme) репозиторий "Б". Склонируйте его к себе ("третий"). Выполните локально несколько коммитов, передайте их в "Б". Приложите ссылку на "Б"

4.* Сделайте еще один клон "Б" - "четвертый". Измените один и тот же файл в локальном репозитории "третий", git push, затем в "четвертом" (те же строки, но другие изменения) - и тоже попытка git push. Если возникнет конфликт - попробуйте объяснить его суть.

*Контрольные вопросы:*

1. Как отправить изменения на удаленный репозиторий?
2. Как влить изменения в локальную ветку из удаленного репозитория?
3. Какой командой можно зафиксировать изменения в репозитории?
4. Как используется Gihub в реальной работе?
5. Опишите принцип коллективной работы с Github

*Список литературы:*

1. Гохберг, Г. С. Информационные технологии : учебник / Г.С. Гохберг, А.В. Зафиевский, А.А. Короткин. - 9-е изд., перераб. и доп. - М. : Академия, 2014. - 240 с.

Белов, В. В. Проектирование информационных систем : учебник / В.В. Белов, В.И. Чистяков ; под ред. В.В. Белова. - М. : Академия, 2013. - 352 с. - (Бакалавриат). - На учебнике гриф: Рек.УМО. - Библиогр.: с. 345-347. - ISBN 978-5-7695-7406-1.

## ЗАДАНИЕ 4. ОЦЕНКА ПО УЧЕБНОЙ И ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ (ПРАКТИЧЕСКОЙ ПОДГОТОВКЕ)

### 4.1. Общие положения

Целью оценки по практической подготовке (учебной и производственной практике) является оценка: 1) профессиональных и общих компетенций; 2) практического опыта и умений.

Оценка по практической подготовке (учебной и производственной практике) выставляется на основании данных аттестационного листа (характеристики профессиональной деятельности обучающегося на практике) с указанием видов работ, выполненных обучающимся во время практики, их объема, качества выполнения в соответствии с технологией и требованиями организации, в которой проходила практика.

### 4.2. Результаты и основные показатели оценки результата практической подготовки (учебной практики)

Таблица 3

| **Результаты (освоенные профессиональные компетенции)** | **Основные показатели оценки результата** | **Документ, подтверждающий качество выполнения работ** |
|---|---|---|
| ПК 1.1 Проектировать информационные ресурсы | **Владеть навыками**<br>– проектирования информационных систем и ресурсов;<br>**Уметь**<br>– применять методы системного анализа;<br>– интерпретировать бизнес-требования заказчика для разработки концептуальной модели информационного ресурса;<br>**Знать**<br>– основы теории системного анализа и построения концептуальных моделей информационных ресурсов средствами графических нотаций;<br>– понятия, классификацию информационных систем и ресурсов;<br>– этапы, принципы и особенности проектирования информационных систем и ресурсов;<br>– архитектуру информационных систем и ресурсов;<br>– модели процесса разработки информационных систем и ресурсов;<br>– принципы проектирования пользовательских интерфейсов; | Дневник учебной практики, аттестационный лист |
| ПК 1.2 Разрабатывать интерфейсы пользователя | **Владеть навыками**<br>– разработки прототипов пользовательских интерфейсов;<br>**Уметь** | Дневник учебной практики, аттестационный лист |

| | | |
|---|---|---|
| | − разрабатывать концептуальную модель информационного ресурса средствами графических нотаций;<br>− разрабатывать прототипы пользовательских интерфейсов с использованием UI/UX подхода<br>**Знать**<br>− основы теории системного анализа и построения концептуальных моделей информационных ресурсов средствами графических нотаций;<br>− понятия, классификацию информационных систем и ресурсов;<br>− этапы, принципы и особенности проектирования информационных систем и ресурсов;<br>− архитектуру информационных систем и ресурсов;<br>− модели процесса разработки информационных систем и ресурсов;<br>− принципы проектирования пользовательских интерфейсов;<br>− элементы управления пользовательского интерфейса;<br>− модели процесса разработки информационных систем и ресурсов;<br>− современные методики тестирования информационных ресурсов.<br>− принцип устройства систем хранения версий кода.<br>− Интерфейсы управления системами хранения версий кода. | |
| ПК 1.3 Интегрировать программный код в соответствующую инфраструктуру | **Владеть навыками**<br>− разработки тестовых сценариев программного средства;<br>**Уметь**<br>− разрабатывать прототипы пользовательских интерфейсов с использованием UI/UX подхода;<br>**Знать**<br>− элементы управления пользовательского интерфейса;<br>− модели процесса разработки информационных систем и ресурсов; | Дневник учебной практики, аттестационный лист |
| ПК 1.4 Использовать систему контроля версий в процессе коллективной (параллельной) разработки | **Владеть навыками**<br>− работы с системой контроля версий, в том числе при коллективной разработке.<br>**Уметь**<br>− создавать, клонирования, развития репозиториев хранения кода;<br>− создавать ветки репозитория и управления изменениями кода; | Дневник учебной практики, аттестационный лист |

| | | |
|---|---|---|
| | − решать конфликты версий кода.<br>**Знать**<br>− принцип устройства систем хранения версий кода.<br>− Интерфейсы управления системами хранения версий кода. | |
| ПК 1.5 Выполнять процедуры тестирования программного кода. | **Владеть навыками**<br>− тестирования информационного ресурса в соответствии с планом тестирования;<br>− документирования результатов тестирования;<br>**Уметь**<br>− выбирать и комбинировать техники тестирования информационных ресурсов;<br>− тестировать информационный ресурс с использованием тест-планов;<br>− применять инструменты подготовки тестовых данных;<br>− работать с инструментами подготовки тестовых данных;<br>− создавать отчет по результатам тестирования.<br>**Знать**<br>− модели процесса разработки информационных систем и ресурсов;<br>− современные методики тестирования информационных ресурсов. | |

Таблица 4

| **Результаты (освоенные общие компетенции)** | **Основные показатели оценки результата** | **Документ, подтверждающий качество выполнения** |
|---|---|---|
| ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам. | **Уметь:**<br>- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;<br>- анализировать задачу и/или проблему и выделять её составные части; определять этапы решения задачи;<br>- выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы;<br>- составить план действия; определить необходимые ресурсы;<br>- владеть актуальными методами работы в профессиональной и смежных сферах;<br>- реализовать составленный план;<br>- оценивать результат и последствия своих действий (самостоятельно или с помощью наставника); | Дневник учебной практики, аттестационный лист |

| | | |
|---|---|---|
| | **Знать:**<br>- актуальный профессиональный и социальный контекст, в котором приходится работать и жить; основные источники информации и ресурсы для решения задач и проблем в профессиональном и/или социальном контексте;<br>- алгоритмы выполнения работ в профессиональной и смежных областях; методы работы в профессиональной и смежных сферах; структуру плана для решения задач;<br>- порядок оценки результатов решения задач профессиональной деятельности. | |
| ОК 02. Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности. | **Уметь:**<br>- определять задачи для поиска информации;<br>- определять необходимые источники информации; планировать процесс поиска;<br>- структурировать получаемую информацию;<br>- выделять наиболее значимое в перечне информации; оценивать практическую значимость результатов поиска; оформлять результаты поиска.<br>**Знать:**<br>- номенклатура информационных источников, применяемых в профессиональной деятельности;<br>- приемы структурирования информации;<br>- формат оформления результатов поиска информации. | Дневник учебной практики, аттестационный лист |
| ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях. | **Уметь:**<br>- определять актуальность нормативно-правовой документации в профессиональной деятельности;<br>- применять современную научную профессиональную терминологию;<br>- определять и выстраивать траектории профессионального развития и самообразования.<br>**Знать:**<br>- содержание актуальной нормативно-правовой документации;<br>- современная научная и профессиональная терминология; возможные траектории профессионального развития и самообразования. | Дневник учебной практики, аттестационный лист |
| ОК 04. Эффективно взаимодействовать и работать в коллективе и команде. | **Уметь:**<br>- организовывать работу коллектива и команды;<br>- взаимодействовать с коллегами, | Дневник учебной практики, аттестационный лист |

| | | |
|---|---|---|
| | руководством, клиентами в ходе профессиональной деятельности.<br>**Знать:**<br>- психологические основы деятельности коллектива, психологические особенности личности;<br>- основы проектной деятельности. | |
| ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста. | **Уметь:**<br>- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке, проявлять толерантность в рабочем коллективе.<br>**Знать:**<br>- особенности социального и культурного контекста;<br>- правила оформления документов и построения устных сообщений. | Дневник учебной практики, аттестационный лист |
| ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках. | **Уметь:**<br>- определять актуальность нормативно-правовой документации в профессиональной деятельности;<br>- применять современную научную профессиональную терминологию;<br>- определять и выстраивать траектории профессионального развития и самообразования.<br>**Знать:**<br>- содержание актуальной нормативно-правовой документации;<br>- современная научная и профессиональная терминология;<br>- возможные траектории профессионального развития и самообразования. | Дневник учебной практики, аттестационный лист |

**4.3 Перечень заданий для оценки освоения учебной практики**

| **Перечень заданий** |
|---|
| Проанализировать организацию заказчика и составить графическую нотацию для представления бизнес процессов в нескольких моделях (AS IS / TO BI) |
| С помощью специализированного ПО или веб-сервисов разработать сайтмэп и прототипы веб приложения учитывая UI/UX. |
| **МДК.01.02. Разработка интерфейсов пользователя** |
| Выполнить тестирование и составить отчет с результатом выбранного веб - ресурса |
| **МДК.01.03. Тестирование информационных ресурсов** |
| Работа с системой контроля версий, в том числе с использованием коллективной разработке |
| Настройка и тестирование сайта организации |
| **Дифференцированный зачет** |

**Уровень освоения профессиональных компетенций**

| ВПД | Профессиональные компетенции (с указанием видов работ) | Результаты освоения (освоен/не освоен) |
|---|---|---|
| Проектирование и разработка информационных ресурсов | ПК 1.1 Проектировать информационные ресурсы<br>Проанализировать организацию заказчика и составить графическую нотацию для представления бизнес процессов в нескольких моделях (AS IS / TO BI) | |
| | ПК 1.2 Разрабатывать интерфейсы пользователя<br>С помощью специализированного ПО или веб-сервисов разработать сайтмэп и прототипы веб приложения учитывая UI/UX | |
| | ПК 1.3 Интегрировать программный код в соответствующую инфраструктуру<br>Выполнить тестирование и составить отчет с результатом выбранного веб - ресурса | |
| | ПК 1.4 Использовать систему контроля версий в процессе коллективной (параллельной) разработки<br>Работа с системой контроля версий, в том числе с использованием коллективной разработке | |
| | ПК 1.5 Выполнять процедуры тестирования программного кода.<br>Настройка и тестирование сайта | |

Характеристика учебной и профессиональной деятельности обучающегося во время учебной/производственной практики *(дополнительная характеристика дается в произвольной форме)*_____
_____
_____
_____

«_____» _____ 20____ г.

Руководитель практики от Техникума _____/_____/
М.П.

### *4.5. Задания для текущего контроля по УП.01.01 Учебная практика.*

**ИНСТРУКЦИОННО - ТЕХНОЛОГИЧЕСКАЯ КАРТА**
на выполнение практической работы №1

**ТЕМА:** Описание организации. Анализ бизнес процессов
**НАИМЕНОВАНИЕ РАБОТЫ:** Разработка структуры и дизайна сайта.
**ЦЕЛЬ РАБОТЫ:** Проектирования и тестирования (исследования) веб-сайтов с помощью редактора HTML-страниц.
**ФОРМИРУЕМЫЕ КОМПЕТЕНЦИИ:** ОК4,ОК5, ОК6.
**ПРИОБРЕТАЕМЫЕ УМЕНИЯ И НАВЫКИ:** освоить основные теги создания HTML-документа.
**НОРМА ВРЕМЕНИ:** 90 минут
**ОСНАЩЕНИЕ РАБОЧЕГО МЕСТА:** ПК.
**ОСОБЫЕ ПРАВИЛА ПО ТЕХНИКЕ БЕЗОПАСНОСТИ НА РАБОЧЕМ МЕСТЕ:** проведен инструктаж по техники безопасности в компьютерном классе.
**ЛИТЕРАТУРА:** ***Компьютерные сети. Принципы, технологии, протоколы. Олифер В. Г., Олифер Н.А. Спб.: Питер,2008, стр. 16-19***

**ВОПРОСЫ ПРИ ДОПУСКЕ:**

1. Перечислите наиболее распространенные виды логических структур веб-сайтов.
2. Почему в именах файлов нежелательно использование символов русского алфавита.

**МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ:**

Разработка структуры сайта необходима для:

- создания четкой и логичной схемы навигации;
- организации простой технологии внесения изменений при редактировании сайта.

Для достижения этих целей процесс создания структуры принято рассматривать в двух аспектах. Фактически проектируются две структуры: логическая и физическая. Логическая структура определяет, в какой последовательности материалы будут доступны пользователю, какие ссылки следует выбирать для доступа к информации, размещенной на сайте. Хорошо продуманная логическая структура гарантирует, что на поиск необходимых данных будет затрачено меньше времени, и что они всегда будут найдены. Для создания полноценной логической структуры достаточно следовать нескольким простым правилам:

- любой документ сайта должен оказываться доступным не более чем с помощью трех переходов с главной страницы сайта;
- все навигационные элементы должны отображаться сразу после загрузки страницы;
- все внутренние связи должны быть двунаправленными, то есть позволять перемещаться между документами в обоих направлениях;
- с любой страницы должен быть предусмотрен возврат на главную страницу сайта;
- названия рубрик и распределение материала между ними должно быть понятным каждому посетителю сайта.
- если сайт имеет более одного уровня навигации, он обязательно должен содержать навигационную карту.
- для удобства посетителей, каждый сайт должен иметь простую, четкую и логичную схему навигации.

Пример логической структуры веб-сайта показан на рис. 1.

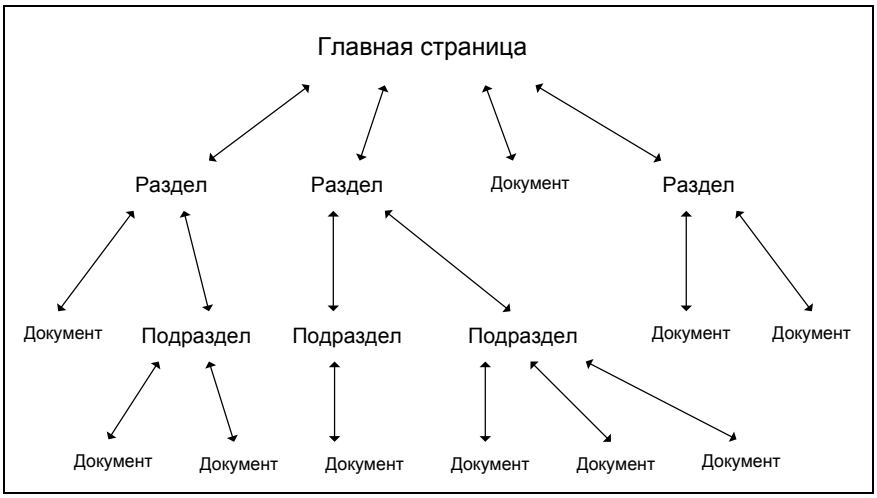



*Рис. 1. Логическая структура сайта*

Существует несколько видов логических структур сайта. Самая простая из них – линейная. В ней страницы следуют одна за другой, и пользователь должен просматривать их последовательно, как слайд-шоу. Недостатков у такой структуры достаточно много. Как следствие, область ее применения ограничена. Она может использоваться на сайтах-презентациях и в онлайновых учебных пособиях. Самый простой вариант сайта с линейной структурой – набор страниц, с каждой из которых есть ссылка на следующую и на предыдущую. Однако и здесь начинающие авторы допускают ошибки. Дело в том, что на каждой странице обязательно должны присутствовать соответствующий заголовок и ссылка на первую страницу. В противном случае посетители, попавшие в середину сайта – например, в результате обращения поисковой системы, не смогут сориентироваться в ситуации, и покинут проект разочарованными. Кроме того, полезно показывать общее число страниц в отображаемом документе и выделять номер той из них, которая воспроизводится в данный момент времени.

Следующим вариантом структуры сайта является линейная структура с альтернативами и вариантами. Ее основой остается простое линейное размещение страниц. Однако на сайтах, построенных по данному принципу, посетители могут проявить некоторую инициативу, позволяющую облегчить поиск нужной информации. Под альтернативами здесь понимается выбор между двумя ветвями перемещения. Чаще всего подобная структура используется для сбора информации о посетителе. Примером может служить процесс регистрации клиента на сайте фирмы, оказывающей определенные услуги. Работа всегда начинается со стартовой страницы. Однако затем частным лицам предлагается ввести одну информацию, а представителям организаций – другую. Возможно, ветви альтернатив в дальнейшем вновь смыкаются, и все пользователи попадают на одну и ту же страницу.

Третий вариант носит название линейной структуры с ответвлениями. Это тоже контролируемая структура перемещения по ресурсам, наглядный аналог которой – дорога с ответвляющимися от нее тупиковыми тропами. Иными словами, посетитель последовательно переходит с одной страницы на другую. Если информация, размещенная на любой из них, его заинтересовала, и он желает ознакомиться с ней подробнее, то ему

предоставляется возможность перейти на ответвление, а потом вернуться обратно на основную "дорогу". Главное преимущество такой структуры состоит в том, что к ней легко перейти с обычного линейного размещения страниц. Часто бывает, что однажды созданный сайт перестает удовлетворять возросшим требованиям, а глобальная переделка по тем или иным причинам нежелательна. В этом случае автор может без затруднений расширить свой проект. И его сайт не утратит четкости логических связей.

Следующий вариант – древовидная структура (рис. 1) – универсальный и в большинстве приложений наиболее предпочтительный способ размещения страниц. Она хорошо зарекомендовала себя для создания практически любых типов сайтов. Пользователь, попадая на главную страницу, оказывается перед выбором, в каком направлении двигаться дальше. После перехода в нужный раздел, он выбирает необходимый подраздел, затем пункт (параграф) и т.д. У древовидной структуры достаточно много достоинств, однако и она не лишена недостатков. Остановимся на главном из них.

В древовидной структуре очень сложно соблюдать баланс между глубиной и шириной. Формальные критерии либо тривиальны, либо рассчитаны на узкие группы пользователей. Успех зависит от опыта, интуиции и квалификации автора. Если "дерево" сайта будет расти только вглубь, то посетителям, чтобы найти необходимую информацию, придется загрузить и просмотреть слишком много страниц. Естественно, это занимает много времени и раздражает. Если же создается очень широкая древовидная структура, то приходится тратить время по другой причине – для выбора нужной ветви поиска. Использование древовидной структуры сайта вынуждает постоянно следить за ее разрастанием и придерживаться золотой середины. Качество работы автора зависит от его мастерства, а не от соблюдения формальных правил.

Еще одним вариантом является решетчатая структура. Эта структура заметно сложнее рассмотренных ранее. В ней все страницы также размещаются в различных ветвях. Однако пользователю дается возможность перемещаться по ним не только вертикально (вверх и вниз), но и горизонтально (то есть между ветвями разных уровней). Используется «решетка» в основном только в каталогах. При этом перемещение между ветками на глубинных уровнях осуществляется с помощью ссылок на рубрики в других разделах.

Использование решетчатой структуры в других проектах считается нецелесообразным. Во-первых, она относительно сложна в реализации. Во-вторых, обращаться с «решеткой» нужно с очень большой осторожностью. В противном случае в схеме возникают непредусмотренные связи, и поиск информации приводит к непредсказуемым результатам.

Физическая структура не влияет на просмотр страниц посетителями и служит в основном для удобства создателя сайта при его редактировании, позволяя легко найти нужный файл (документ). При проектировании физической структуры разработчик может ненадолго забыть о пользователях и немного подумать об удобствах сопровождения сайта. Обычно распределяют отдельные папки для каждого из его разделов и подразделов. Внутри папок также создают отдельные папки для вспомогательных изображений, отдельные папки для мультимедиа-информации (музыка, видео, и т.п.) – то есть файлы

сайта разделяются по функциональным признакам. В простом статическом сайте можно четко определить несколько групп файлов:

- страницы сайта, представляющие собой HTML-файлы;
- таблицы стилей;
- клиентские скрипты;
- графические файлы, используемые в дизайне сайта;
- файлы для копирования посетителями.

Особое место занимает вопрос, связанный с выбором имен файлов. Логично и понятно названный файл позволяет сэкономить время при обновлении сайта. Кроме того, при передаче поддержки сайта другому сотруднику, будет гораздо легче объяснить, где находится та или иная информация. Возможно, на первых порах это кажется неважным, однако опыт показывает, что разбираться в структуре неудачно именованного сайта довольно затруднительно.

Прежде всего – не следует называть файлы безликими именами, такими как page1.htm, 123.htm и т.п. Необходимо, чтобы при взгляде на список файлов сразу становилось понятно, что в них содержится. Другими словами, называть файлы следует исходя из содержательного смысла документа. Если страница излагает общую информацию о компании, то принято называть ее *about.htm*, а страницу с контактной информацией – *contacts.htm*.

В некоторых случаях для группы файлов могут использоваться похожие наименования, состоящие из базового имени и цифры. Например, если вы периодически пишете статьи на своем сайте, то файлы удобно называть последовательно: *article_1.htm, article_2.htm, article_3.htm* и т.д. Заметим, что в качестве разделителя базового имени и цифры используется знак подчеркивания. Он позволяет отделить номер статьи, что способствует быстрому нахождению нужного файла.

При формировании имен файлов можно взять за основу либо русский язык, либо английский. Так как использовать русские символы в именах файлов нельзя (причины этого описаны ниже), то при использовании русского в качестве базового языка необходимо писать имена файлов в транслитерации (русские слова латинскими буквами). В этом случае страницу с описаниями услуг можно назвать, например, *uslugi.htm*, а страницу с информацией об истории фирмы – *istoriya.htm*. Предпочтительно использовать в качестве основного языка английский, потому что в этом случае слова будут корректно индексироваться поисковыми системами, а значит, при корректном запросе по теме вашего сайта релевантность (степень соответствия поисковому запросу) вашей страницы будет выше. Не следует лишь забывать о правильном написании английских слов. Кроме этого, если имена файлов используют английский язык, то человек, который попал на вашу страницу и не знает русский язык, сможет догадаться, о чем идет речь. Выбрав язык для использования в именах файлов, придерживайтесь его в пределах всего сайта. Не допускайте чередования английских и русских имен файлов.

Для зависимых файлов, например, для иллюстраций к какой-либо странице, удобно

использовать следующее правило: *имя графического файла образуется из названия страницы и идентификатора иллюстрации, разделенных знаком подчеркивания*. В качестве идентификатора иллюстрации может использоваться либо порядковый номер появления ее в документе, либо, что предпочтительнее, некоторый идентификатор, позволяющий легко опознать ее. Допустим, что наша статья называется *article_1.htm,* и в ней используются иллюстрации – пусть это будут фотографии сотрудников отдела и схема их взаимного подчинения. Тогда имена графических файлов, образованных согласно данному правилу, будут соответствовать конкретным фамилиям, например: *article_1_ivanov. jpg, article_1_petrov.jpg, article_1_sidorov.jpg,* а иерархия отношений – штатной схеме отдела: *article_1_scheme.gif.*

Для части графических файлов, преимущественно участвующих в создании дизайна сайта, удобно использовать префиксы и суффиксы. Если спроектировано графическое меню сайта, которое подсвечивается при наведении курсора манипулятора мышь, то все графические файлы, формирующие меню, можно предварять префиксом «m_», а к названиям изображений, которые появляются при наведении мышью, добавлять суффикс «_over». Тогда название графического пункта меню, например "О компании", будет состоять из двух файлов – «*m_about.gif»* и «*m_about_over.gif»*.

Префиксы удобно добавлять к таким изображениям, которые могут изменяться в зависимости от типа страницы. Приведем несколько часто используемых префиксов:

- «bg_» (background) – для фоновых изображений;
- «m_» (menu) – для пунктов графического меню;
- «t_» (title) – для графических заголовков;
- «icon_» – для пиктограмм;
- «button_» – для графических кнопок, не являющихся элементами меню.

В качестве «корня» слова, образующего имя файла, удобно использовать название страницы, к которой относится данная графика. В приведенном примере «корнем» выступала страница *about*. Продолжим упражнения с ней и образуем имя файла, используемого как подложки этой страницы – получится имя *bg_about.gif*. Разумеется, имеет смысл это делать, если фоновый рисунок на разных страницах различный. Если же он везде одинаковый, то файл достаточно назвать просто *bg.gif*. Такова основа методики формирования имен файлов.

Перечисленные шаги выполняются с единственной целью – облегчить ориентирование в огромном количестве файлов, из которых состоит любой современный сайт. В результате нетрудно найти нужную html-страницу и все относящиеся к ней иллюстрации, а также понять содержимое графических файлов, даже не заглядывая в них.

При создании имен файлов следует помнить об ограничениях, накладываемых со стороны операционных систем. Например, в формате UNIX-систем index.html и Index.html – разные файлы, а с точки зрения Windows – одни и те же. Нужна осторожность при использовании русских букв в именах файлов – преобразование символов из одной кодировки в другую превратит файл "галерея.htm" в "ЗБМЕТЕС.htm".

Рассмотрим инструментальное средство разработки сайтов, применяемое в лабораторных работах по курсу «Пакеты прикладных программ».

ПОДГОТОВКА К ВЫПОЛНЕНИЮ РАБОТЫ

Подготовить сменный носитель (flash-накопитель) для сохранения результатов работы с целью их дальнейшего использования в работах №№ 2 – 4.

По рекомендованной литературе ознакомиться с принципами работы в среде Microsoft FrontPage.

Ознакомиться с базовыми элементами языка HTML.

Выбрать тематику будущего веб-сайта.

Ознакомиться с дизайном и структурой существующих веб-сайтов.

Подготовить текст теоретических разделов отчета о работе.

Ответить на контрольные вопросы настоящих методических указаний.

## ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

РАЗРАБОТКА ЛОГИЧЕСКОЙ СТРУКТУРЫ ВЕБ-САЙТА. Определите тип и количество информации, которая будет помещена на сайт. В результате должно получиться не менее пяти разделов. К полученным разделам обязательно добавьте разделы «Анкета» и «Гостевая книга».

ВЫБОР ВИДА ЛОГИЧЕСКОЙ СТРУКТУРЫ САЙТА. Выберите вид логической структуры веб-сайта в соответствии с рекомендациями, данными в разделе 2.1. Предпочтительной является древовидная структура сайта. Подобную структуру имеет сайт, описываемый ниже в качестве примера.

ЛОКАЛИЗАЦИЯ ПРОЕКТА. Средствами Windows создайте папку для хранения файлов вашего сайта. Создайте в ней подпапки в соответствии с разработанной физической структурой сайта. При разработке физической структуры рекомендуется создавать отдельную подпапку для каждого раздела (остальные рекомендации сформулированы в разделе 2.1).

ЗАПУСК MICROSOFT FRONTPAGE. Запустите среду создания веб-сайтов Microsoft Frontpage (Пуск – Программы – Microsoft Frontpage).

СОЗДАНИЕ СТРАНИЦ САЙТА. В появившемся окне редактора веб-страниц введите необходимую информацию.

Нажмите правую кнопку мыши и выберите меню «Свойства страницы». Пройдите по всем закладкам появившегося окна и установите все необходимые вам свойства. Особое внимание обратите на последнюю закладку, на которой устанавливаются языковые параметры (рис. 2).

Введите информацию, необходимую для размещения на титульной странице (рис. 3). Сохраните полученный файл.

Создайте новую страницу. Расположите на ней элементы меню сайта в соответствии с созданной логической структурой (рис. 4).

Создайте остальные страницы сайта (кроме анкеты и гостевой книги) в соответствии с разработанной структурой. Наполните их информацией. На одну из страниц поместите крупный блок информации, преимущественно текстовой (рис. .7).

Позаботьтесь о том, чтобы сделать дизайн сайта привлекательным для посетителей, используя предоставленные вам изобразительные средства.

ЗАВЕРШЕНИЕ РАБОТЫ. Сохраните получившиеся страницы и перепишите созданные файлы на заранее подготовленный носитель. Выйдите из Frontpage, выключите компьютер.

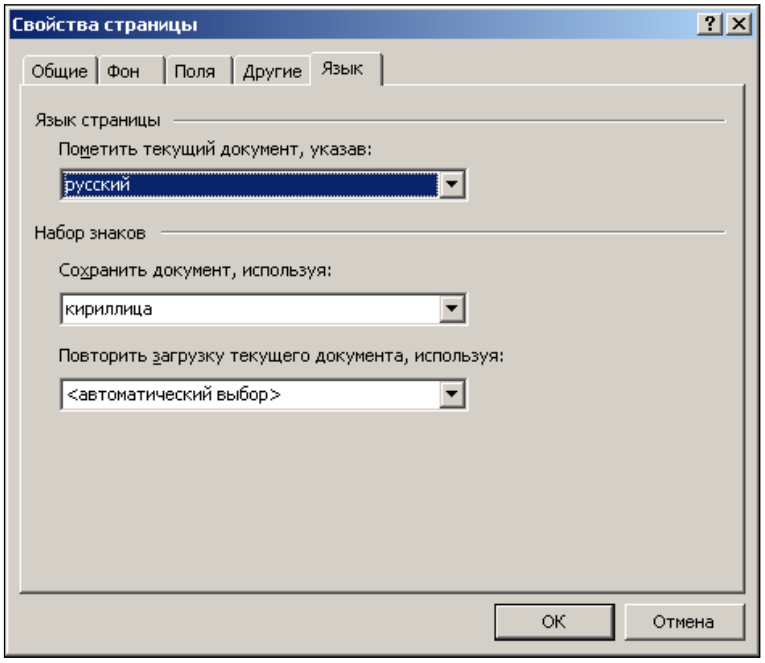



Рис. 2 Установка свойств страницы в редакторе

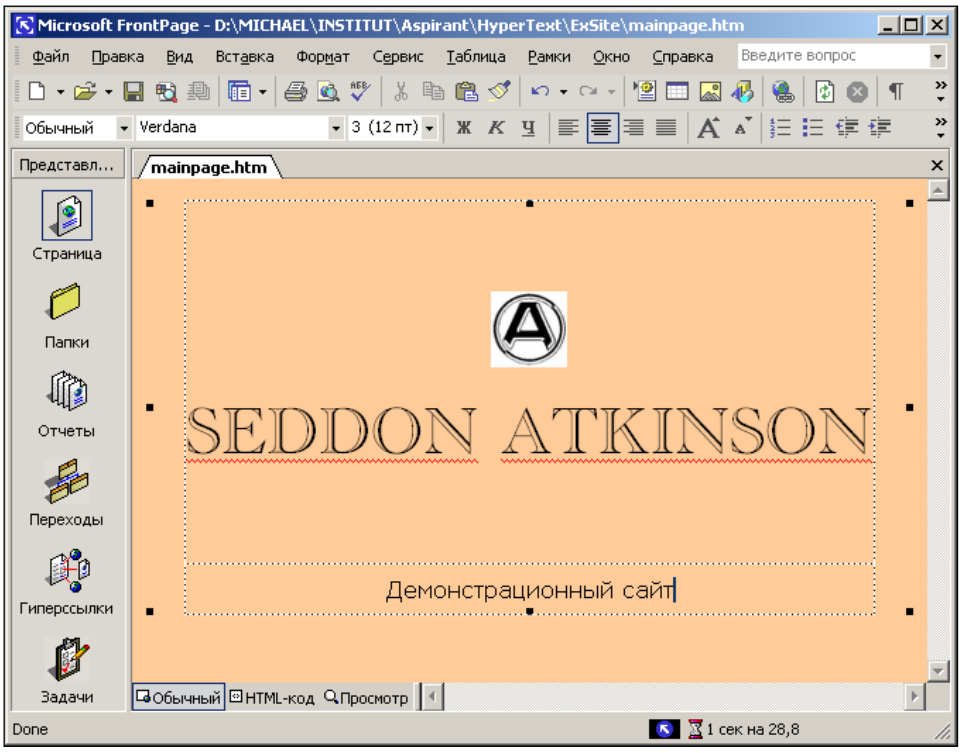
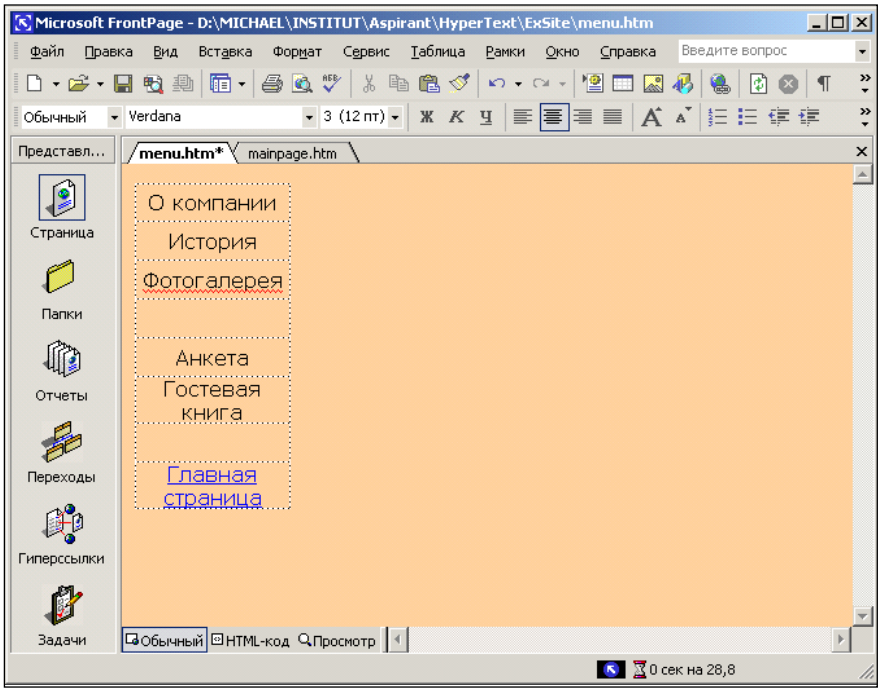



Рис. 3 Формирование титульной страницы




## СОДЕРЖАНИЕ ОТЧЕТА

Отчет о работе должен содержать следующие материалы в соответствии с полученным индивидуальным заданием:

Обоснование выбора структуры сайта.

Схему логической структуры созданного сайта.

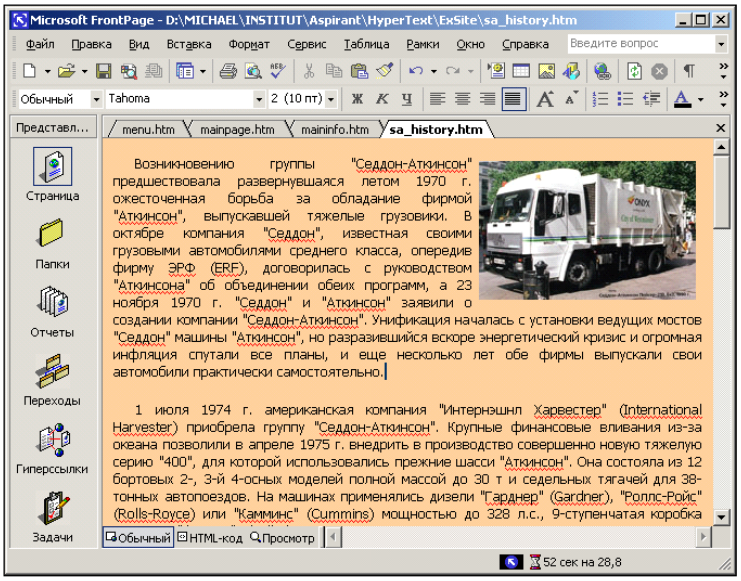
Схему физической структуры созданного сайта.

Словесную постановку задачи лабораторной работы (по материалам индивидуального задания).

Последовательность выполнения работ в среде Microsoft FrontPage для создания сайта.

Формализацию выполненного индивидуального задания (разделение структуры и оформления создаваемых документов).

Ответы на контрольные вопросы и выводы по выполненной работе.




## КОНТРОЛЬНЫЕ ВОПРОСЫ

1. С какой целью (для использования в какой части сайта) мог быть создан файл *bg_about_me.jpg*.
2. Сформулируйте принцип организации, преимущества, недостатки и область применения решетчатой структуры сайта.
3. Укажите достоинства и недостатки пакета Microsoft FrontPage как универсального инструмента для обмена и управления узлами в Web в сравнении с другими средствами редактирования HTML-страниц.
4. Параметры (атрибуты) какого элемента определяются заново при изменении в редакторе цвета фона веб-страницы.
5. Какими атрибутами или какими элементами HTML устанавливаются наклон, толщина, подчеркивание и другие особенности шрифта в тексте и его фрагментах.

## 4.6. Задания для промежуточной аттестации по УП.01.01 и ПП.01.01

### 4.6.1 Результаты и основные показатели оценки результата практической подготовки (производственной практики)

Таблица 5

| Вид деятельности | Код и формулировка компетенции | Показатели освоения компетенции |
|---|---|---|
| *1* | *2* | *3* |

| | | |
|---|---|---|
| ВПД. 1 Проектирование и разработка информационных ресурсов | ПК 1.1.Проектировать информационные ресурсы | **Владеть навыками**<br>− проектирования информационных систем и ресурсов;<br>**Уметь**<br>− применять методы системного анализа;<br>− интерпретировать бизнес-требования заказчика для разработки концептуальной модели информационного ресурса;<br>**Знать**<br>− основы теории системного анализа и построения концептуальных моделей информационных ресурсов средствами графических нотаций;<br>− понятия, классификацию информационных систем и ресурсов;<br>− этапы, принципы и особенности проектирования информационных систем и ресурсов;<br>− архитектуру информационных систем и ресурсов;<br>− модели процесса разработки информационных систем и ресурсов;<br>− принципы проектирования пользовательских интерфейсов; |

| | | |
|---|---|---|
| | ПК 1.2.Разрабатывать интерфейсы пользователя | **Владеть навыками**<br>− разработки прототипов пользовательских интерфейсов;<br>**Уметь**<br>− разрабатывать концептуальную модель информационного ресурса средствами графических нотаций;<br>− разрабатывать прототипы пользовательских интерфейсов с использованием UI/UX подхода<br>**Знать**<br>− основы теории системного анализа и построения концептуальных моделей информационных ресурсов средствами графических нотаций;<br>− понятия, классификацию информационных систем и ресурсов;<br>− этапы, принципы и особенности проектирования информационных систем и ресурсов;<br>− архитектуру информационных систем и ресурсов;<br>− модели процесса разработки информационных систем и ресурсов;<br>− принципы проектирования пользовательских интерфейсов;<br>− элементы управления пользовательского интерфейса;<br>− модели процесса разработки информационных систем и ресурсов;<br>− современные методики тестирования информационных ресурсов.<br>− принцип устройства систем хранения версий кода.<br>− Интерфейсы управления системами хранения версий кода. |
| | ПК 1.3.Интегрировать программный код в соответствующую инфраструктуру | **Владеть навыками**<br>− разработки тестовых сценариев программного средства;<br>**Уметь**<br>− разрабатывать прототипы пользовательских интерфейсов с использованием UI/UX подхода;<br>**Знать**<br>− элементы управления пользовательского интерфейса;<br>− модели процесса разработки информационных систем и ресурсов; |

| | | |
|---|---|---|
| | ПК 1.4. Использовать систему контроля версий в процессе коллективной (параллельной) разработки | **Владеть навыками**<br>− работы с системой контроля версий, в том числе при коллективной разработке.<br>**Уметь**<br>− создавать, клонирования, развития репозиториев хранения кода;<br>− создавать ветки репозитория и управления изменениями кода;<br>− решать конфликты версий кода.<br>**Знать**<br>− принцип устройства систем хранения версий кода.<br>− Интерфейсы управления системами хранения версий кода. |
| | ПК.1.5. Выполнять процедуры тестирования программного кода. | **Владеть навыками**<br>− тестирования информационного ресурса в соответствии с планом тестирования;<br>− документирования результатов тестирования;<br>**Уметь**<br>− выбирать и комбинировать техники тестирования информационных ресурсов;<br>− тестировать информационный ресурс с использованием тест-планов;<br>− применять инструменты подготовки тестовых данных;<br>− работать с инструментами подготовки тестовых данных;<br>− создавать отчет по результатам тестирования.<br>**Знать**<br>− модели процесса разработки информационных систем и ресурсов;<br>− современные методики тестирования информационных ресурсов. |
| | ОК 01. Выбирать способы решения задач профессиональной деятельности применительно к различным контекстам. | **Уметь:**<br>- распознавать задачу и/или проблему в профессиональном и/или социальном контексте;<br>- анализировать задачу и/или проблему и выделять её составные части; определять этапы решения задачи;<br>- выявлять и эффективно искать информацию, необходимую для решения задачи и/или проблемы;<br>- составить план действия; определить необходимые ресурсы;<br>- владеть актуальными методами работы в профессиональной и смежных сферах;<br>- реализовать составленный план; |

| | | |
|---|---|---|
| | | - оценивать результат и последствия своих действий (самостоятельно или с помощью наставника);<br>**Знать:**<br>- актуальный профессиональный и социальный контекст, в котором приходится работать и жить; основные источники информации и ресурсы для решения задач и проблем в профессиональном и/или социальном контексте;<br>- алгоритмы выполнения работ в профессиональной и смежных областях; методы работы в профессиональной и смежных сферах; структуру плана для решения задач;<br>- порядок оценки результатов решения задач профессиональной деятельности. |
| | ОК 02. Использовать современные средства поиска, анализа и интерпретации информации, и информационные технологии для выполнения задач профессиональной деятельности. | **Уметь:**<br>- определять задачи для поиска информации;<br>- определять необходимые источники информации; планировать процесс поиска;<br>- структурировать получаемую информацию;<br>- выделять наиболее значимое в перечне информации; оценивать практическую значимость результатов поиска; оформлять результаты поиска.<br>**Знать:**<br>- номенклатура информационных источников, применяемых в профессиональной деятельности;<br>- приемы структурирования информации;<br>- формат оформления результатов поиска информации. |
| | ОК 03. Планировать и реализовывать собственное профессиональное и личностное развитие, предпринимательскую деятельность в профессиональной сфере, использовать знания по финансовой грамотности в различных жизненных ситуациях. | **Уметь:**<br>- определять актуальность нормативно-правовой документации в профессиональной деятельности;<br>- применять современную научную профессиональную терминологию;<br>- определять и выстраивать траектории профессионального развития и самообразования.<br>**Знать:**<br>- содержание актуальной нормативно-правовой документации;<br>- современная научная и профессиональная терминология; возможные траектории профессионального развития и самообразования. |

| | | |
|---|---|---|
| | ОК 04. Эффективно взаимодействовать и работать в коллективе и команде. | **Уметь:**<br>- организовывать работу коллектива и команды;<br>- взаимодействовать с коллегами, руководством, клиентами в ходе профессиональной деятельности.<br>**Знать:**<br>- психологические основы деятельности коллектива, психологические особенности личности;<br>- основы проектной деятельности. |
| | ОК 05. Осуществлять устную и письменную коммуникацию на государственном языке Российской Федерации с учетом особенностей социального и культурного контекста. | **Уметь:**<br>- грамотно излагать свои мысли и оформлять документы по профессиональной тематике на государственном языке, проявлять толерантность в рабочем коллективе.<br>**Знать:**<br>- особенности социального и культурного контекста;<br>- правила оформления документов и построения устных сообщений. |
| | ОК 09. Пользоваться профессиональной документацией на государственном и иностранном языках. | **Уметь:**<br>- определять актуальность нормативно-правовой документации в профессиональной деятельности;<br>- применять современную научную профессиональную терминологию;<br>- определять и выстраивать траектории профессионального развития и самообразования.<br>**Знать:**<br>- содержание актуальной нормативно-правовой документации;<br>- современная научная и профессиональная терминология;<br>- возможные траектории профессионального развития и самообразования. |

Таблица 6

**4.6.2. Перечень заданий для оценки освоения производственной практики**

| **Перечень заданий** |
|---|
| **МДК.01.01. Проектирование информационных ресурсов** |
| Проанализировать организацию заказчика и составить графическую нотацию для представления бизнес процессов в нескольких моделях (AS IS / TO BI) |
| С помощью специализированного ПО или веб-сервисов разработать сайтмэп и прототипы веб приложения учитывая UI/UX. |
| **МДК.01.02. Разработка интерфейсов пользователя** |
| Выполнить тестирование и составить отчет с результатом выбранного веб - ресурса |
| **МДК.01.03. Тестирование информационных ресурсов** |
| Работа с системой контроля версий, в том числе с использованием коллективной разработке |
| Настройка и тестирование сайта |

| |
|---|
| **дифференцированный зачет** |

**4.6.3.Форма аттестационного листа по производственной практике**

**Уровень освоения профессиональных компетенций**

| **ВПД** | **Профессиональные компетенции (с указанием видов работ)** | **Результаты освоения (освоен/не освоен)** |
|---|---|---|
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

Характеристика учебной и профессиональной деятельности обучающегося во время учебной/производственной практики (дополнительная характеристика дается в произвольной форме)_____
_____
_____
_____

«_____» _____ 20___ г.

Руководитель практики от Предприятия _____/_____/
МП (при наличии)

Руководитель практики от Техникума _____/_____/
М.П.

**4.6.4 Задания для промежуточной аттестации по УП.01.01 и ПП.01.01**

**Задания для комплексного дифференцированного зачета по учебной и производственной практике**

**ЗАДАНИЕ 6** ***(практического характера)***

***Для выполнения заданий необходимо на рабочем столе компьютера открыть папку «ФИО». Для выполнения задания необходимо создать текстовый документ в этой же папке и выполнить действия в зависимости от варианта.***

## I блок заданий

1. Ознакомиться с теоретическим материалом.

2. Проанализировать предметную область.

2.1 Исходные данные (требования пользователей):

Необходимо разработать систему для клуба любителей скачек. Основная информация предметной области это участвующие в скачках лошади, а так же полный список лошадей задействованных в различных соревнованиях. Так же необходимо иметь доступ к

данным о владельцах лошадей, для рассылки необходимой информации или уведомлении их об организации очередного соревнования. Участвующие в соревновании жокеи заранее выбираются владельцами лошадей, при желании владельца лошади на соревновании могут быть зарегистрированы несколько жокеев (при этом они могут быть зарегистрированы для участия на одной лошади или для всех лошадей конкретного владельца). Так же хранимой информацией должна быть информация о самих соревнованиях.

2.2 Ответить на вопросы для выявления информационных объектов предметной области и связей между ними (вопросы представлены в теоретическом материале).

2.3 Построить модель предметной области и спроектировать схему базы данных (при разработке БД разрешается вводить собственные ограничения на данные предметной области, *согласовав их с преподавателем*).

3. Используя online разработать функциональную схему работы информационной системы в рамках данной предметной области, охватывающую все сферу деятельности клуба, таких как:

3.1 регистрация новых лошадей/владельцев/жокеев в системе;

3.2 организация и проведение соревнований;

3.3 уведомление участников о планирующемся мероприятии/соревновании;

3.4 сбор членских взносов;

3.5 обеспечение документооборота;

3.6 исключение участников в связи с нарушениями «внутренних правил»;

3.7 продолжите перечисление самостоятельно.

Имеется возможность применить графический элемент «предопределенный процесс» при разработке функциональной схемы с использованием детализации этих процессов в виде отдельных функциональных схем.

4. Реализовать разработанную схему базы данных в СУБД «Access».

5. По аналогии реализовать отдельную БД (в соответствии с индивидуальным заданием) и функциональную схему информационной системы для работы с ней.

**II блок заданий**

1. Даны два числа. Найти их сумму, разность, произведение, а также частное от деления первого числа на второе.
2. Даны два числа. Найти среднее арифметическое и среднее геометрическое их модулей.
3. Составить программу решения линейного уравнения ax + b = 0 (a≠0).
4. Составить программу обмена значениями двух переменных величин.
5. Определить максимальное и минимальное значения из двух различных вещественных чисел.
6. Даны вещественные числа a, b, c (a≠0). Выяснить, имеет ли уравнение ax2 + bx + c = 0.
7. Определить, является ли число a делителем числа b? А наоборот?
8. Составить программу с получением на ПК правильных ответов.

$$y=\begin{cases} e^{x} & when\ x\le 0 \\ x\ln x & when\ 0<x\le 1 \\ x+1 & when\ x>1 \end{cases}$$

9. ставить программу для вычисления и печати таблицы умножения на 12.
10. Вычислить значение выражения y.

$$Y=\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\frac{1}{5^2}+\cdots+\frac{1}{15^2}$$

**III блок заданий**

**Техническое задание на разработку сайта.**

ТЗ — это документ, который описывает будущий проект детально и полностью. Чем детальней он будет, тем точнее будет реализована задумка и тем меньше конфликтов и спорных ситуаций в ходе выполнения проекта будет возникать, ведь абсолютно любую вещь можно сделать по-разному. На него можно ссылаться, если что-то не выполнено или выполнено не так или допущены другие ошибки. Перед началом работ заказчик обычно в тезисном виде описывает будущий проект или заполняет бриф, а исполнитель формализует все эти требования и пожелания, при необходимости, предлагает корректировки. При этом заказчику необходимо убедиться, что все его хотения зафиксированы в тех задании.

В соответствии с заполненным вами ранее брифом, требуется разработать ТЗ (в папке задания ознакомится с примером разработанного ТЗ).

Оно должно состоять из основных разделов (на ваше усмотрение разделы могут быть дополнены в соответствии с данными из брифа):

1. Общие требования (назначение и цели разрабатываемого сайта)
2. Требования к дизайну сайта.
3. Требования к функциональности сайта
4. Требования к содержимому сайта
5. Согласование и подписи сторон

**Разработка БРИФа**

1. Написать для заказчика вопросы БРИФа на разработку программного обеспечения - это может быть мобильное приложение, разработка базы данных и приложения для работы с ним или свой вариант (разработка сайта не допускается).

Вопросы должны быть тестовыми и должны быть:

- задания закрытой формы;
- свободное изложение/дополнение;
- установление соответствия;
- установление последовательности;

Использовать как минимум 1 вопрос каждого типа. Минимум должно быть 20 вопросов.

2. Предложить другой группе ответить на вопросы вашего брифа.

3. Основываясь на полученных ответах БРИФа, сформулировать примерное техническое задание в свободной форме (считать, что заказчик изначально настроен на повременную форму оплату).

**IV блок заданий**

Программа выбора авиакомпании и рейса – и бронирование билета на этот рейс на указанный пользователем день.

1. Создать базу данных Аэропорт, данные которой будут храниться:
   - в двух текстовых файлах (для каждой таблицы): Компания (ID, название, год основания, рейтинг) и Рейс (№, ID, город вылета, город прилета, время вылета, время прилета).
2. В новом проекте Visual Studio создать форму. Добавить на неё

элемент «Меню». Продумать начальное приветственное изображение данной формы.

3. Меню будет содержать два пункта – «Аэропорт» (выпадающий список – «Добавление данных», «Просмотр данных») и «Заказ билетов».

4. При выборе пункта меню «Аэропорт» - «Добавление данных»: открывается новая форма с двумя вкладками «Компания» и «Рейс». На каждой вкладке имеется возможность просмотреть все данные по соответствующей таблице, а так же добавить (удалить) данные (строки).

5. При выборе пункта меню «Аэропорт» - «Просмотр данных»: из файлов на форму в **два элемента вывода данных** загружается список авиакомпаний и данные по рейсам:

- при нажатии ЛКМ на ячейку таблицы с авиакомпаниями – в 1 текстовое поле заносится указанное пользователем название компании.
- при нажатии ЛКМ на ячейку таблицы с рейсами – в 2 текстовое поле заносится указанный номер рейса.
- если тестовые поля (см.выше) не заполнены, то при переходе в пункт меню

  «Аэропорт» - «Заказ билетов», появляется сообщение с ошибкой.

6. Пункт меню «Аэропорт» - «Заказ билетов», будет содержать данные, указанные пользователем в п.5 - авиакомпания и рейс.

7. Здесь же располагается календарь(-ри) с помощью которых можно выбрать дату вылета и дату возвращения, а так же две кнопки – «забронировать» и «оплатить».

8. Выбирая даты, необходимо предусмотреть, что бы дата вылета была меньше даты возвращения (иначе вывести предупреждение).

9. Используя элементы формы предоставить пользователю возможность рассчитать стоимость полета. Базовую стоимость полета установить самостоятельно (и отобразить пользователю). Она будет актуальна для будних дней. Если полет бронируется на выходные, то стоимость перелета увеличивается на 10%.

10. Кнопка «оплатить» будет доступна только после нажатия кнопки «забронировать» в течение одной минуты. Если за это время оплата не совершена (не нажата кнопка), то все изменения, которые внес пользователь, на данной форме сбрасываются до начального значения (по умолчанию).

11. Продумать интерфейс программы и диалог с пользователем, для удобства его работы.

12. Добавьте на форму элемент ProgressBar для отчета времени между нажатиями кнопок

«забронировать» и «оплатить»

## V блок заданий

В соответствии с представленным теоретическим материалом выполнить задания.

Задание 1. На основании технического задания и анализа требований к программному обеспечению разработать функциональную схему программного обеспечения и диаграмму деятельности (с не менее чем 4-мя объектами «деятельность») по своему варианту задания. Под схемой грамотно описать их работу (общий объем текста описания >= 1500 знаков).

Задание 2. На основании функциональной схемы программного обеспечения определить какая информация для ПО будет являться входной и выходной. Для этого:

- используя Ramus составить нулевой уровень декомпозиции;
- описать вх/вых данные;
- заполнить таблицу с атрибутами: «тип информации» (вх/вых), «наименование информации», «тип данных», «примерный размер передаваемых данных» (с описанием расчета значений).

Задание 3. По описаниям вариантов деятельности из вашего варианта задания построить прототипы программного обеспечения (Visual Studio) с описанием «невидимого функционала» (составить как минимум четыре полноценные рабочие формы программы – если работаете в паре, каждый разрабатывает по две формы).

Задание 4. Провести анализ эргономичности графического интерфейса пользователя по прототипам, созданным в предыдущей работе. Составить таблицу соответствия (несоответствия) разработанного пользовательского интерфейса требованиям стандартного графического интерфейса пользователя. Подготовить рекомендации по исправлению выявленных нарушений (каждый из пары вместе работающих студентов анализирует 2 формы своего сокурсника).

Задание 5. На языке программирования составить алгоритмы, реализующие взаимодействие между формами приложения (использовать заготовку прототипа): переходы, передачу данных.

Задание 6. Разработать отдельные процедуры, реализующей контроль входной информации:

- для текстовых данных;
- для числовых данных;
- для ввода дат.

Задание 7. Разработать главную форму программы с использованием контекстного, и главного меню - ContextMenuStrip и MenuStrip.

## ЗАДАНИЕ №7

## 6. Задание для проведения промежуточной аттестации (комплексный экзамен по МДК.01.01 Разработка программных модулей и МДК.01.02 Поддержка и тестирование программных модулей

**Билеты для проведения комплексного экзамена по МДК.01.01 Технические средства сбора, обработки и хранения текстовой информации и МДК.01.02 Порядок оформления и компоновки технической документации**

1. Основные определения раздела «проектирование и дизайн информационных систем»
2. Классификация информационных систем.
3. Обеспечивающие подсистемы АИС
4. Жизненный цикл программного продукта.
5. Стандарты ISO к процессам жизненного цикла.
6. Группы процессов разработки ПО?
7. Классические модели жизненного цикла.
8. Каскадная модель ЖЦ.
9. V-образная модель ЖЦ.
10. Модель быстрой разработки модель ЖЦ.
11. Спиральную модель ЖЦ.
12. Этапы анализа предметной области.
13. Методы сбора материалов исследования.
14. Функциональный и объектно-ориентированный подход сбора материалов обследования.
15. Диаграммы действий.
16. Классификация и характеристики CASE – систем.
17. Модель IDEF0.
18. Диаграммы декомпозиции и диаграммы дерева узлов
19. Программное средство структурного моделирования процессов RAMUS.
20. Модель DFD.
21. Экспертные системы и системы реального времени.
22. Метрики и их использование в разработке ПП.
23. Модель управления качеством.
24. Система стандартизации и сертификации качества продукции
25. Виды угроз информационной безопасности.

26. Методология анализа защищенности информационной системы.
27. Основные требования к моделям предметных областей.
28. Реинжиниринг бизнес-процессов.
29. Требования к разработке пользовательского интерфейса.
30. Требования к графической части интерфейса.
31. Основные документы на разработку ИС.
32. Стандарты ЕСПД и ЕСКД при разработке документации.
33. Самодокументирующиеся программы.
34. Назначение проектной, маркетинговой документации?

**Практическая часть для экзамена**

1. Построить диаграмму прецедентов, описывающую процесс разработки программного продукта.
2. Построить диаграмму прецедентов описывающей работу приемной комиссии.
3. Построить диаграмму объектов, подходящую для разработки программного продукта.
4. Построить диаграмму последовательностей, описывающую работу цветочного магазина.
5. Построить диаграмму взаимодействия, описывающую работу цветочного магазина.
6. Построить диаграмму состояний, описывающую работу приемной комиссии.
7. Построить диаграмму активности, описывающую процесс приготовления ужина.
8. Разработать прототип системы, позволяющей выбрать и оформить туристическую путевку.
9. В среде Ramus создать диаграмму A-0. Для контекстного процесса ИЗГОТОВЛЕНИЕ МЕБЕЛИ определите необходимую информацию:
   - ВХОД - сырьё;
   - УПРАВЛЕНИЕ – чертежи, производственные инструкции, инструкции по технике безопасности (нормативные документы);
   - МЕХАНИЗМЫ – персонал, производственное оборудование;
   - ВЫХОД – готовая мебель.
10. Откройте среду Visual Studio C# 2010 Express: Создайте Новый проект – Форма

Подключите БД BD-00 из сетевой папки студенты – экзамен разработка ИС к проекту Выведите данные полей «Артику» и «Цена» из таблицы Ассортименте на форму Добавьте на форму компонент BindingNavigator
Измените свойство bindingSource для компонента BindingNavigator

Создайте кнопку на форме и текстовое поле, в котором будет вычисляться скидка 10%

11. Скопировать из папки «Сеть – Студенты – Экзамен ПИ-049» базу данных computer. На форме задания 1 вывести таблицу Laptop.
12. Реализовать программу, которая для введенного в текстовое поле числа N вычисляет: 1+2+3…+N.
13. Используя методологию IDEF0 в ramus описать диаграмму деятельности кафетерия (2 уровня декомпозиции)
14. Скопировать из папки «Сеть – Студенты – Экзамен ПИ-049» базу данных computer. На форме в Visual Studio вывести таблицу Product.
15. Привести в соответствие элементы систем (система-элемент-цель):

| | **Система** | **Элементы системы** | **Главная цель системы** |
|---|---|---|---|
| **1** | Фирма | Электронные и электромеханические элементы, линии связи и др. | Производство профессиональной информации |
| **2** | Компьютер | Компьютеры, компьютерные сети, люди, информационное и программное обеспечение | Передача информации |
| **3** | Телекоммуникационная система | Люди, оборудование, материалы, здания и др. | Обработка данных |
| **4** | Информационная система | Компьютеры, модемы, кабели, сетевое программное обеспечение и др. | Производство товаров |

16. Привести в соответствие процессы в ИС:

| | **1** | **2** |
|---|---|---|
| 1 | Ввод информации | представление потребителям или передача в другую систему |
| 2 | Обработка информации | Информация, переработанная людьми данной организации для коррекции входной информации |
| 3 | Вывод информации | из внешних или внутренних источников |

| | | |
|---|---|---|
| 4 | Обратная связь | представление информации в удобном виде |

**Условия проведения:** Кабинет Информатика

**Критерии оценивания:**

Отметка «отлично» ставится, если:

знания отличаются глубиной и содержательностью, дается полный исчерпывающий ответ, как на основные вопросы билета, так и на дополнительные:

- студент свободно владеет научными понятиями;
- студент способен к интеграции знаний по определенной теме, структурированию ответа, к анализу положений существующих теорий, научных школ, направлений по вопросу билета;
- логично и доказательно раскрывает проблему, предложенную в билете;
- ответ не содержит фактических ошибок и характеризуется глубиной, полнотой, уверенностью студента;
- ответ иллюстрируется примерами, в том числе из собственной практики;
- студент демонстрирует умение вести диалог и вступать в научную дискуссию.

Отметка «хорошо» ставится, если:

знания имеют достаточный содержательный уровень, однако отличаются слабой структурированностью; раскрыто содержание билета, имеются неточности при ответе на дополнительные вопросы:

- в ответе имеют место несущественные фактические ошибки, которые студент способен исправить самостоятельно, благодаря наводящему вопросу;
- недостаточно раскрыта проблема по одному из вопросов билета;
- недостаточно логично построено изложение вопроса;
- ответ прозвучал недостаточно уверенно;
- студент не смог показать способность к интеграции и адаптации знаний или теории и практики.

Отметка «удовлетворительно» ставится, если:

знания имеют фрагментарный характер, отличаются поверхностностью и малой содержательностью содержание билета раскрыто слабо, имеются неточности при ответе на основные вопросы билета:

- программные материал в основном излагается, но допущены фактические ошибки;
- ответ носит репродуктивный характер;
- студент не может обосновать закономерности и принципы, объяснить факты;
- нарушена логика изложения, отсутствует осмысленность представляемого материала;
- у студента отсутствуют представления о межпредметных связях.
- Отметка «неудовлетворительно» ставится, если:
- обнаружено незнание или непонимание студентом сущностной части социальной психологии;
- допускаются существенные фактические ошибки, которые студент не может исправить самостоятельно;

На большую часть дополнительных вопросов по содержанию экзамена студент затрудняется дать ответ или не дает верных ответов.

**ЗАДАНИЕ №8**

## 7. Задания для промежуточной аттестации по оценке профессионального модуля

### Задания для экзамена по модулю

Оцениваемые компетенции:
*ПК 1.1.Проектировать информационные ресурсы*

**Задание 1.** По изложенным в теоретическом материале критериям заказчика составить и сформулировать следующие виды документов:

1) требования к системе,
2) концепцию создания информационной системы,
3) техническое задание.

Оцениваемые компетенции:
*ПК 1.2.Разрабатывать интерфейсы пользователя*
*ПК 1.3.Интегрировать программный код в соответствующую инфраструктуру*

**Задание 2.** Реализовать процедуры первого модуля системы в соответствии с изложенными требованиями заказчика (описание функционала системы представлено в теоретическом материале).

Оцениваемые компетенции:
*ПК 1.4. Использовать систему контроля версий в процессе коллективной (параллельной) разработки*

**Задание 3.** Разработать чек-лист и тест-кейс для проверки правильного функционирования созданного в задании 2 модуля.

Оцениваемые компетенции:
*ПК.1.5. Выполнять процедуры тестирования программного кода.*

**Задание 4.** В соответствии с результатами проведенного тестирования в задании 3, провести оценку разработанного модуля и составить техническую документацию для ее дальнейшей модернизации.

**7.1. Пакет экзаменатора**

**Условия выполнения задания:**

1. Место выполнения задания: Лаборатория Информатики и информационно-коммуникационных технологий.
2. Максимальное время выполнения задания: 90 мин.

| **ПАКЕТ ЭКЗАМЕНАТОРА** | | |
|---|---|---|
| *Теоретическое задание*<br>*Практическое задание* | | |
| **Результаты освоения**<br>(объекты оценки) | **Критерии оценки результата**<br>(в соответствии с разделом 1 «Паспорт комплекта контрольно-оценочных средств) | **Отметка о выполнении** |
| ПК 1.1 Проектировать информационные ресурсы | **Владеть навыками**<br>− проектирования информационных систем и ресурсов;<br>**Уметь**<br>− применять методы системного анализа;<br>− интерпретировать бизнес-требования заказчика для разработки концептуальной модели информационного ресурса;<br>**Знать**<br>− основы теории системного анализа и построения концептуальных моделей информационных ресурсов средствами графических нотаций;<br>− понятия, классификацию информационных систем и ресурсов;<br>− этапы, принципы и особенности проектирования информационных систем и ресурсов;<br>− архитектуру информационных систем и ресурсов;<br>− модели процесса разработки информационных систем и ресурсов;<br>− принципы проектирования пользовательских интерфейсов; | балльная оценка |
| ПК 1.2 Разрабатывать интерфейсы пользователя | **Владеть навыками**<br>− разработки прототипов пользовательских интерфейсов;<br>**Уметь**<br>− разрабатывать концептуальную модель информационного ресурса средствами графических нотаций;<br>− разрабатывать прототипы пользовательских интерфейсов с использованием UI/UX подхода<br>**Знать**<br>− основы теории системного анализа и построения концептуальных моделей информационных ресурсов средствами графических нотаций;<br>− понятия, классификацию информационных систем и ресурсов; | балльная оценка |

| | | |
|---|---|---|
| | − этапы, принципы и особенности проектирования информационных систем и ресурсов;<br>− архитектуру информационных систем и ресурсов;<br>− модели процесса разработки информационных систем и ресурсов;<br>− принципы проектирования пользовательских интерфейсов;<br>− элементы управления пользовательского интерфейса;<br>− модели процесса разработки информационных систем и ресурсов;<br>− современные методики тестирования информационных ресурсов.<br>− принцип устройства систем хранения версий кода.<br>− Интерфейсы управления системами хранения версий кода. | |
| ПК 1.3 Интегрировать программный код в соответствующую инфраструктуру | **Владеть навыками**<br>− разработки тестовых сценариев программного средства;<br>**Уметь**<br>− разрабатывать прототипы пользовательских интерфейсов с использованием UI/UX подхода;<br>**Знать**<br>− элементы управления пользовательского интерфейса;<br>− модели процесса разработки информационных систем и ресурсов; | балльная оценка |
| ПК 1.4 Использовать систему контроля версий в процессе коллективной (параллельной) разработки | **Владеть навыками**<br>− работы с системой контроля версий, в том числе при коллективной разработке.<br>**Уметь**<br>− создавать, клонирования, развития репозиториев хранения кода;<br>− создавать ветки репозитория и управления изменениями кода;<br>− решать конфликты версий кода.<br>**Знать**<br>− принцип устройства систем хранения версий кода.<br>− Интерфейсы управления системами хранения версий кода. | балльная оценка |
| ПК 1.5 Выполнять процедуры тестирования программного кода. | **Владеть навыками**<br>− тестирования информационного ресурса в соответствии с планом тестирования;<br>− документирования результатов тестирования;<br>**Уметь**<br>− выбирать и комбинировать техники тестирования информационных ресурсов;<br>− тестировать информационный ресурс с использованием тест-планов;<br>− применять инструменты подготовки тестовых данных;<br>− работать с инструментами подготовки тестовых данных;<br>− создавать отчет по результатам тестирования. | балльная оценка |

| | | |
|---|---|---|
| | **Знать**<br>− модели процесса разработки информационных систем и ресурсов;<br>− современные методики тестирования информационных ресурсов. | |