Отчёт по лабораторной работе № 4

Веретенников Дмитрий Олегович НКАбд-01-24

Содержание

# 1 Цель работы

Получение навыков правильной работы с репозиториями git.

# 2 Задание

1. Установка git-flow
2. Установка Node.js
3. Настройка Node.js
4. Общепринятые коммиты
5. Создание репозитория git
6. Работа с репозиторием git

# 3 Теоретическое введение

Gitflow Workflow опубликована и популяризована Винсентом Дриссеном. Gitflow Workflow предполагает выстраивание строгой модели ветвления с учётом выпуска проекта. Данная модель отлично подходит для организации рабочего процесса на основе релизов. Работа по модели Gitflow включает создание отдельной ветки для исправлений ошибок в рабочей среде. Последовательность действий при работе по модели Gitflow: Из ветки master создаётся ветка develop. Из ветки develop создаётся ветка release. Из ветки develop создаются ветки feature. Когда работа над веткой feature завершена, она сливается с веткой develop. Когда работа над веткой релиза release завершена, она сливается в ветки develop и master. Если в master обнаружена проблема, из master создаётся ветка hotfix. Когда работа над веткой исправления hotfix завершена, она сливается в ветки develop и master.

Процесс работы с Gitflow Основные ветки (master) и ветки разработки (develop)

Для фиксации истории проекта в рамках этого процесса вместо одной ветки master используются две ветки. В ветке master хранится официальная история релиза, а ветка develop предназначена для объединения всех функций. Кроме того, для удобства рекомендуется присваивать всем коммитам в ветке master номер версии. При использовании библиотеки расширений git-flow нужно инициализировать структуру в существующем репозитории:

git flow init Для github параметр Version tag prefix следует установить в v. После этого проверьте, на какой ветке Вы находитесь:

git branch Функциональные ветки (feature)

Под каждую новую функцию должна быть отведена собственная ветка, которую можно отправлять в центральный репозиторий для создания резервной копии или совместной работы команды. Ветки feature создаются не на основе master, а на основе develop. Когда работа над функцией завершается, соответствующая ветка сливается обратно с веткой develop. Функции не следует отправлять напрямую в ветку master. Как правило, ветки feature создаются на основе последней ветки develop. Создание функциональной ветки

Создадим новую функциональную ветку:

git flow feature start feature\_branch Далее работаем как обычно.

Окончание работы с функциональной веткой

По завершении работы над функцией следует объединить ветку feature\_branch с develop:

git flow feature finish feature\_branch Ветки выпуска (release)

Когда в ветке develop оказывается достаточно функций для выпуска, из ветки develop создаётся ветка release. Создание этой ветки запускает следующий цикл выпуска, и с этого момента новые функции добавить больше нельзя — допускается лишь отладка, создание документации и решение других задач. Когда подготовка релиза завершается, ветка release сливается с master и ей присваивается номер версии. После нужно выполнить слияние с веткой develop, в которой с момента создания ветки релиза могли возникнуть изменения. Благодаря тому, что для подготовки выпусков используется специальная ветка, одна команда может дорабатывать текущий выпуск, в то время как другая команда продолжает работу над функциями для следующего. Создать новую ветку release можно с помощью следующей команды:

git flow release start 1.0.0 Для завершения работы на ветке release используются следующие команды:

git flow release finish 1.0.0 Ветки исправления (hotfix)

Ветки поддержки или ветки hotfix используются для быстрого внесения исправлений в рабочие релизы. Они создаются от ветки master. Это единственная ветка, которая должна быть создана непосредственно от master. Как только исправление завершено, ветку следует объединить с master и develop. Ветка master должна быть помечена обновлённым номером версии. Наличие специальной ветки для исправления ошибок позволяет команде решать проблемы, не прерывая остальную часть рабочего процесса и не ожидая следующего цикла релиза. Ветку hotfix можно создать с помощью следующих команд:

git flow hotfix start hotfix\_branch По завершении работы ветка hotfix объединяется с master и develop:

git flow hotfix finish hotfix\_branch

# 4 Выполнение лабораторной работы

Устанавливаем git-flow из коллекции репозиториев Copr (рис. 1).

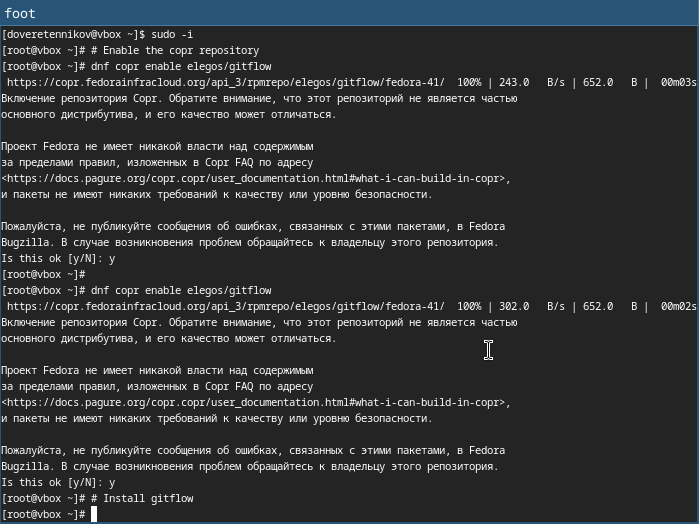


Рис. 1: Процесс подготовки к установке

Установка git-flow (рис. 2).

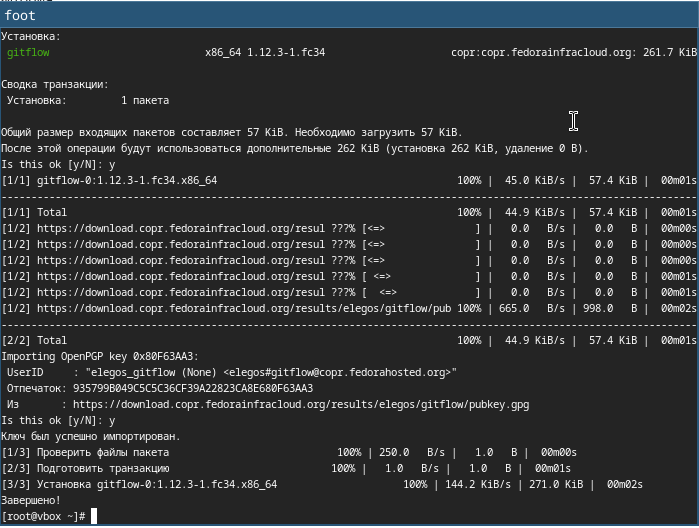


Рис. 2: Процесс установки

Устанавливаем Nodejs и pnpm, далее настраиваем Nodejs (рис. 3).

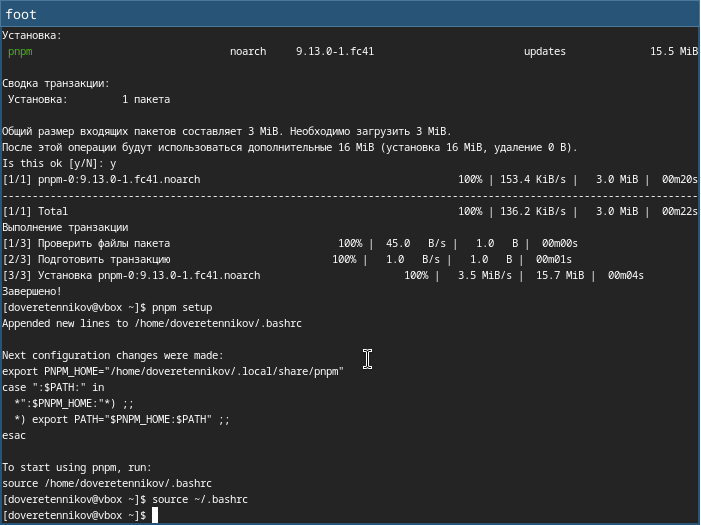


Рис. 3: Процесс установки и настройки

Воспользуемся программой для помощи в форматировании коммитов (рис. 4).

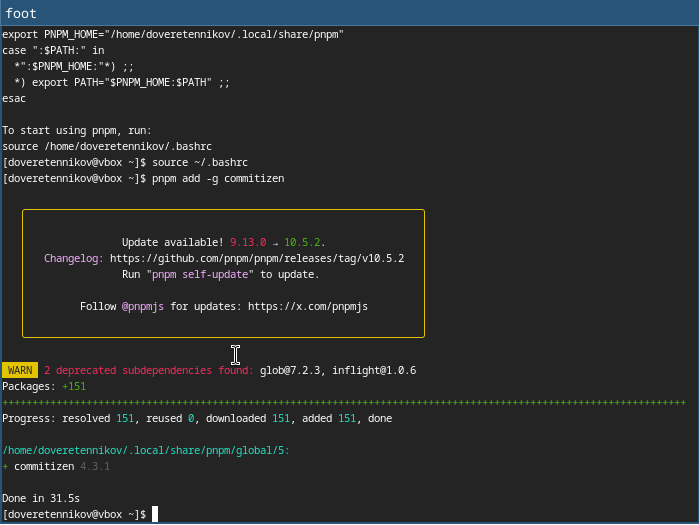


Рис. 4: Программа

Воспользуемся программой для помощи в создании логов (рис. 5).

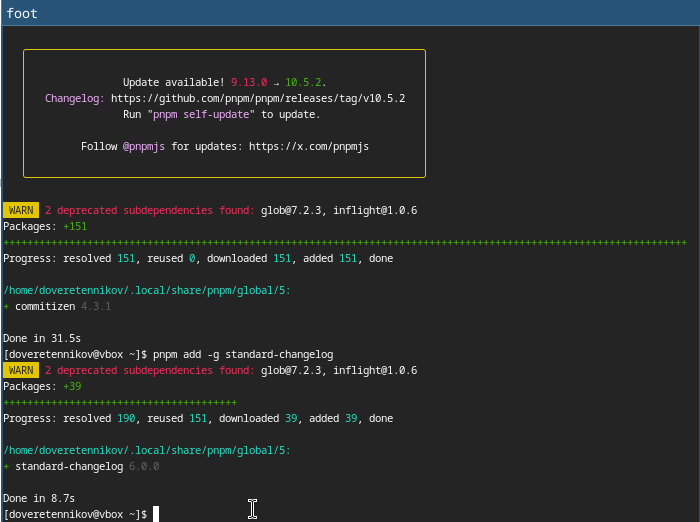


Рис. 5: Программа

Создаем репозиторий, делаем первый коммит и выкладываем на гитхаб (рис. 6).

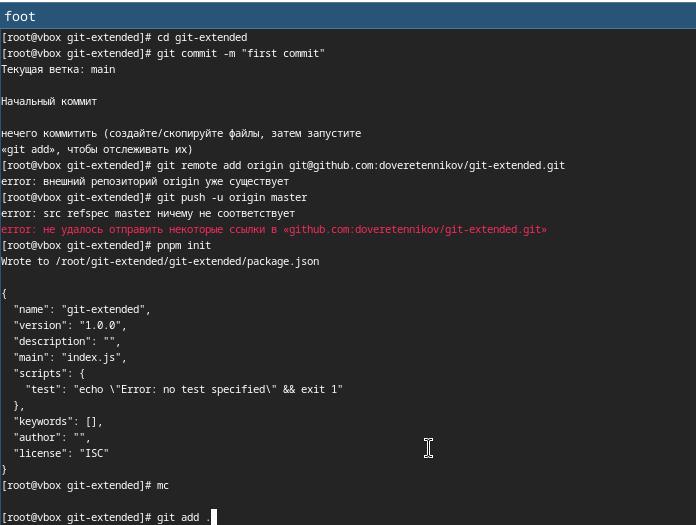


Рис. 6: Отправка коммита

Сконфигурим формат коммитов. Для этого добавим в файл package.json команду для формирования коммитов (рис. 7).

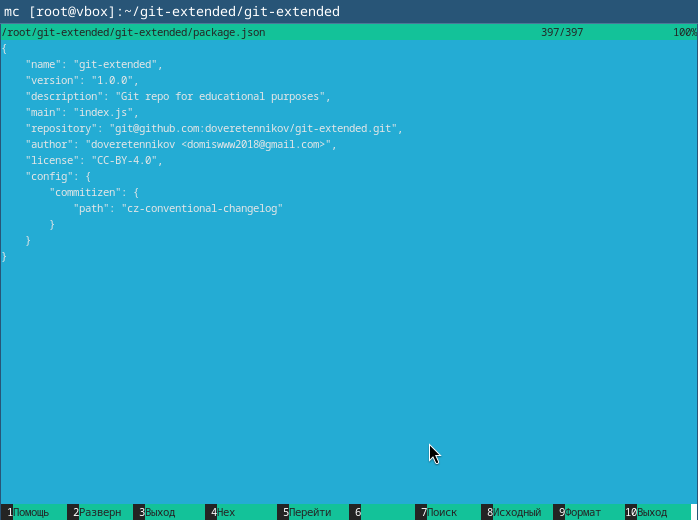


Рис. 7: Исправленный файл package.json

Добавим новые файлы, выполним коммит, отправим на github (рис. 8).

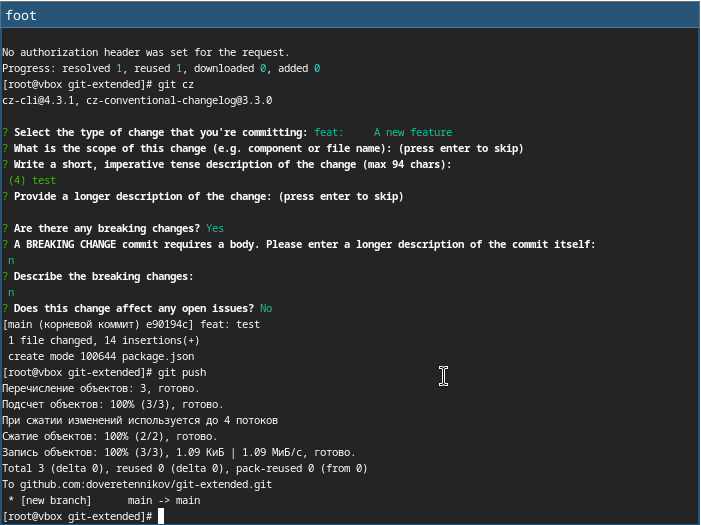


Рис. 8: Выполнение коммита

Инициализируем git-flow (рис. 9).

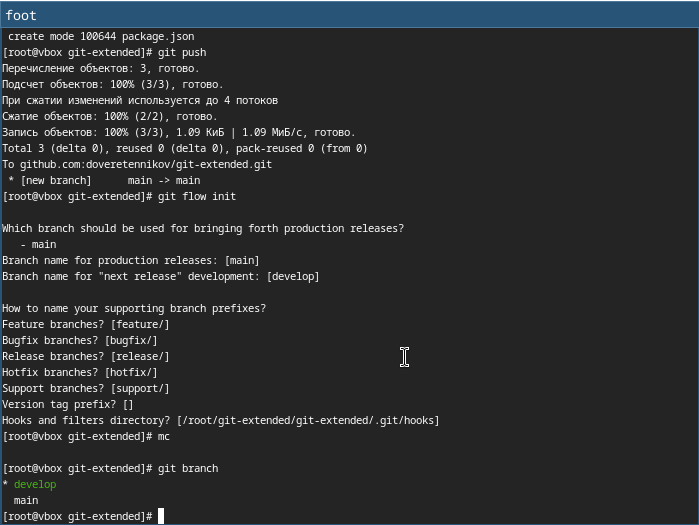


Рис. 9: Выполнение коммита

Загружаем весь репозиторий в хранилище, установим внешнюю ветку как вышестоящую для этой ветки, создадим релиз с версией 1.0.0 (рис. 10).

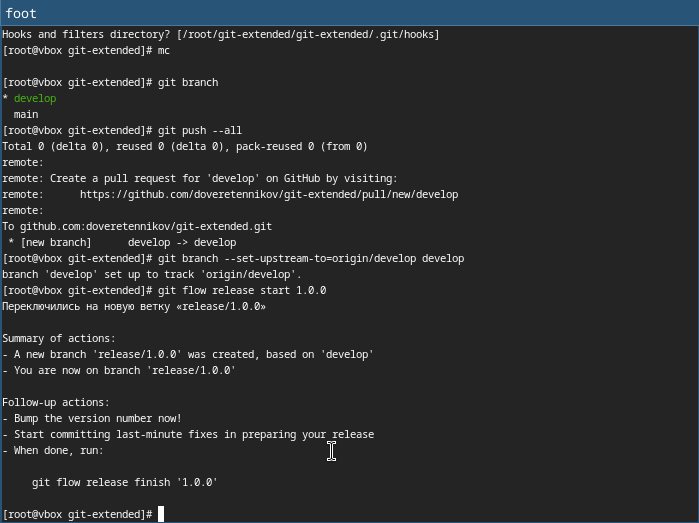


Рис. 10: Загрузка репозитория, создание релиза

Создаем журнал изменений, добавляем журнал изменений в индекс, заливаем релизную ветку в основную ветку (рис. 11).

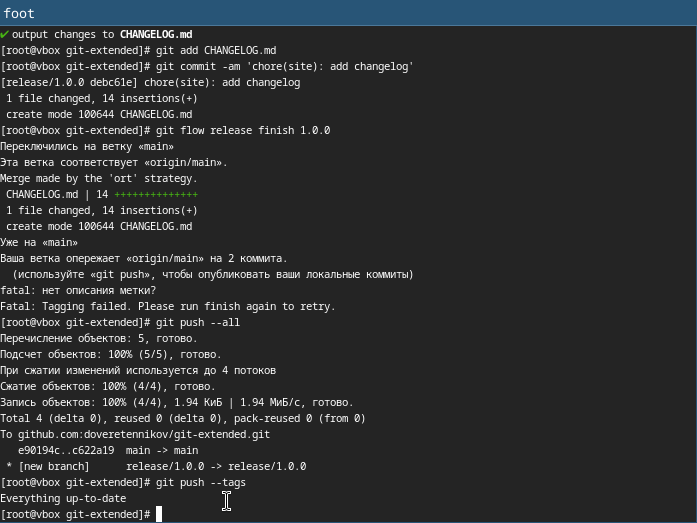


Рис. 11: Создание журнала, добавление журнала, залитие релизной ветки

Создаем ветку для новой функциональности, по окончании разработки новой функциональности объединяем ветку feature\_branch c develop (рис. 12).

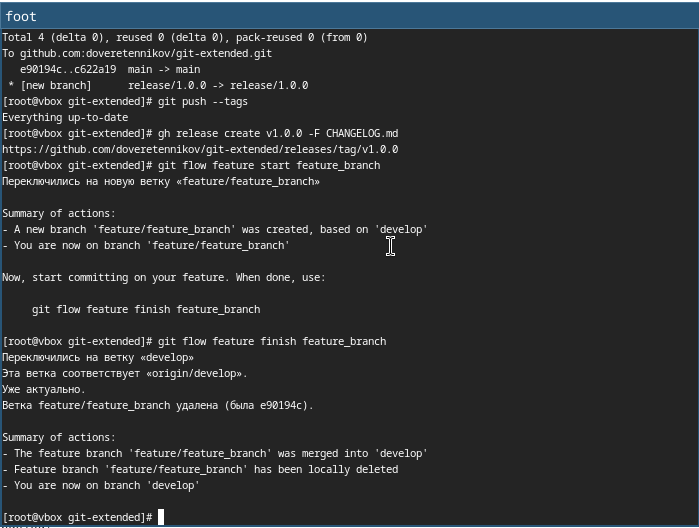


Рис. 12: Создаем и объединяем ветки

Обновите номер версии в файле package.json. Установите её в 1.2.3 (рис. 13).

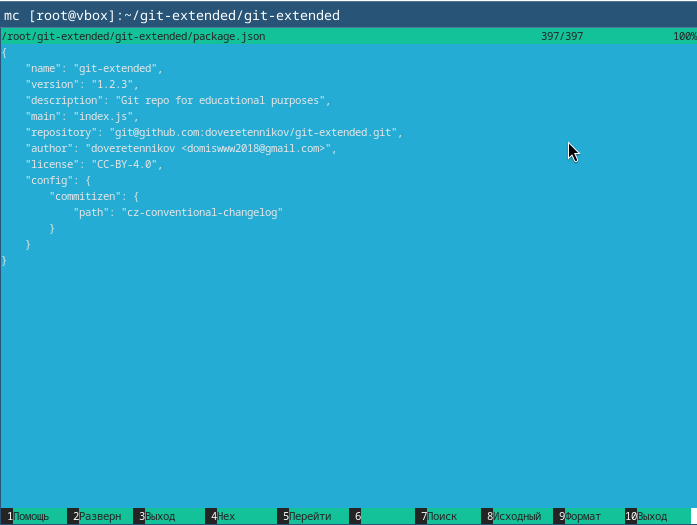


Рис. 13: Обновленный номер версии в файле

Создадем релиз с версией 1.2.3, создадим журнал изменений (рис. 14).

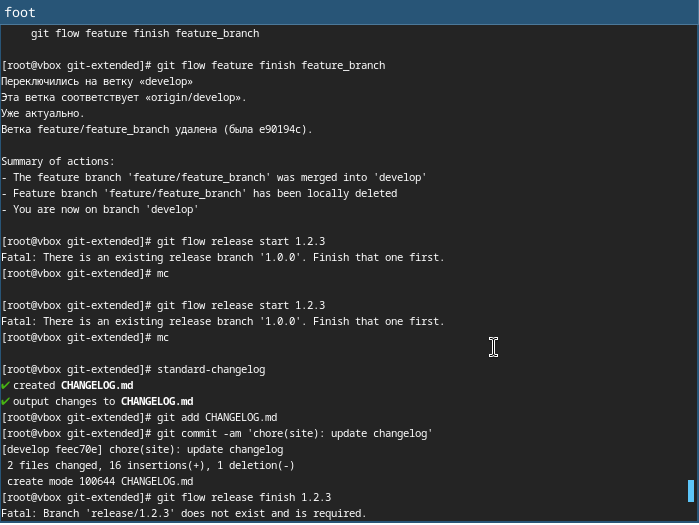


Рис. 14: Создание релиза, создание журнала

Добавляем журнал изменений в индекс, заливаем релизную ветку в основную ветку (рис. 15).

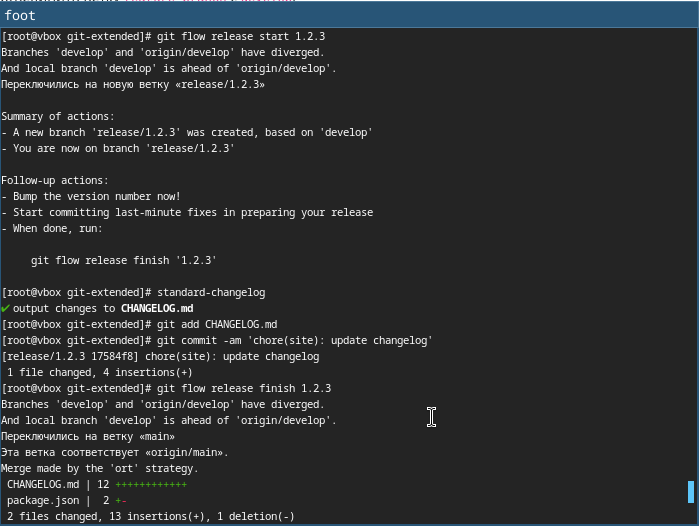


Рис. 15: Создание релиза, создание журнала

Отправляем данные на Github (рис. 16).

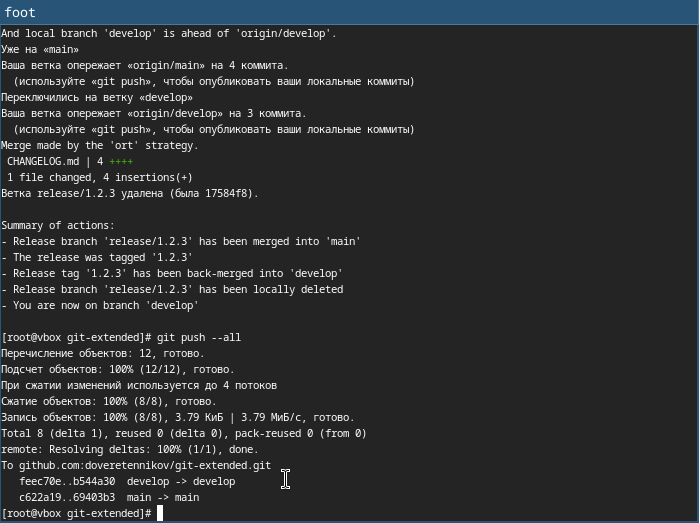


Рис. 16: Отправка данных на Github

# 5 Выводы

После выполнения данной лабораторной работы я получил навыки правильной работы с репозиториями git.

# Список литературы

Курс Архитектура компьютеров и операционные системы. Раздел “Операционные системы” Лабораторная работа № 4 ТУИС РУДН.