

ФГБОУ ВО ПензенскИЙ государственнЫЙ технологическИЙ УНИВЕРСИТЕТ

Кафедра «Прикладная информатика»

Дисциплина «Программирование»

ОТЧЕТ

О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ 1

по теме:

## Система контроля версий GIT

Выполнил:

студент гр. 18СН1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Г. Дружинина

подпись, дата

студент гр. 18СН1 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ К.В. Васильев

подпись, дата

Принял:

Ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.С. Юранов

подпись, дата

Пенза 2019

**Цель лабораторной работы**:получить понимание необходимости инструментов контроля версий (VCS — version control system), изучить основы локальной работы с VCS Git.

**Теоретические основы**

Система контроля версий или система управления версиями - это программа, которая позволяет фиксировать и отслеживать изменения исходного кода **программных проектов**. При этом, сам программный проект представляет собой дерево каталогов, то есть набор файлов и подкаталогов, хранящихся в **каталоге проекта**. Файлы и подкаталоги, которые находятся в каталоге проекта называются **рабочим деревом проекта** (work tree). В дереве проекта, как правило, находится первичный набор исходных файлов, на основе которых выполняется сборка программного проекта. Минимальный набор первичных исходных файлов программного проекта называется **исходным кодом** (source code) проекта. Файлы и подкаталоги дерева проекта, хранящиеся в системе контроля версий, называются **отслеживаемыми файлами** (tracking files). В противовес всем другим файлам, которые могут также оказаться в дереве проекта (неотслеживаемые файлы). Например, промежуточные файлы, генерируемые компилятором или новые файлы с исходным кодом, не добавленные пока в систему контроля версий. Задачами системы контроля версий являются следующие:

* **фиксация изменений** для отслеживаемых, добавляемых или удаляемых файлов исходного кода в рабочем дереве проекта;
* **хранение истории** зафиксированных изменений исходного кода;
* **сравнение и поиск** ранее зафиксированных изменений исходного кода;
* **объединение** наработок и экспериментальных измений ведущихся различными разработчиками над одним исходным кодом.

Цепочка зафиксированных изменений исходного кода называется **историей программного проекта**. История программного проекта хранится в **репозитории программного проекта**. Любая работа над исходным кодом программного проекта выполнятся в рамках репозитория. Репозиторий в **системе контроля версий** Git хранится в каталоге с именем .git, который находится в корневом каталоге рабочего дерева проекта.

Рассмотрим работу с программой Git на примере проекта Visual Studio.

В первую очередь необходимо скачать и установить Git на локальный компьютер. Можно воспользоваться Portable версией (<https://github.com/git-for-windows/git/releases/download/v2.14.1.windows.1/PortableGit-2.14.1-32-bit.7z.exe>).

Следующим шагом запустим оболочку Git Bash (Рисунок 1)

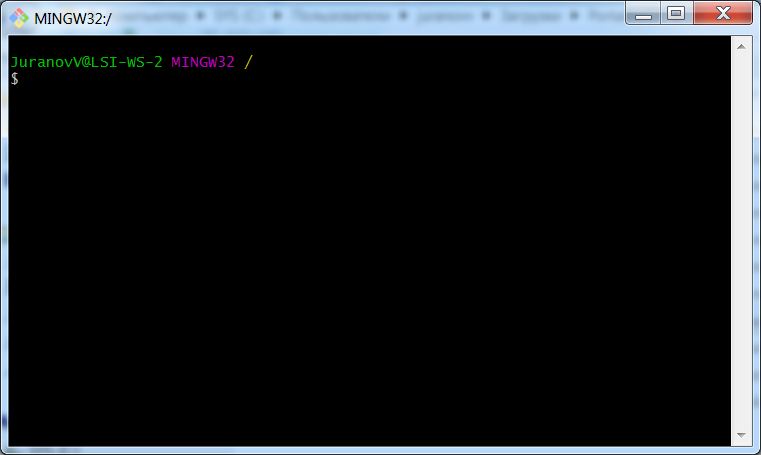
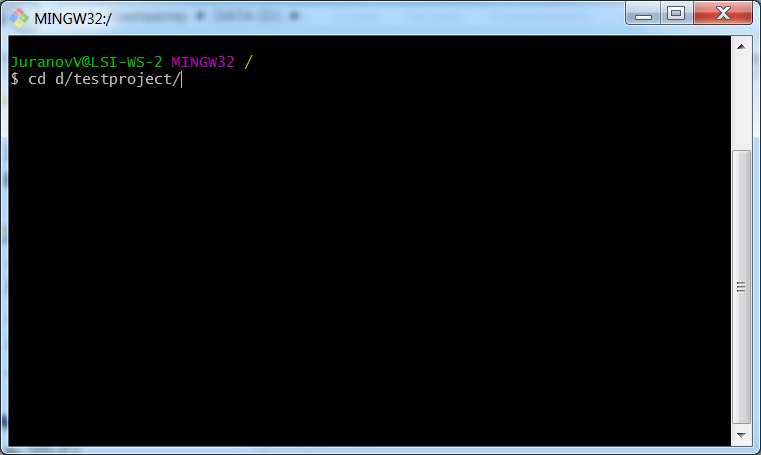
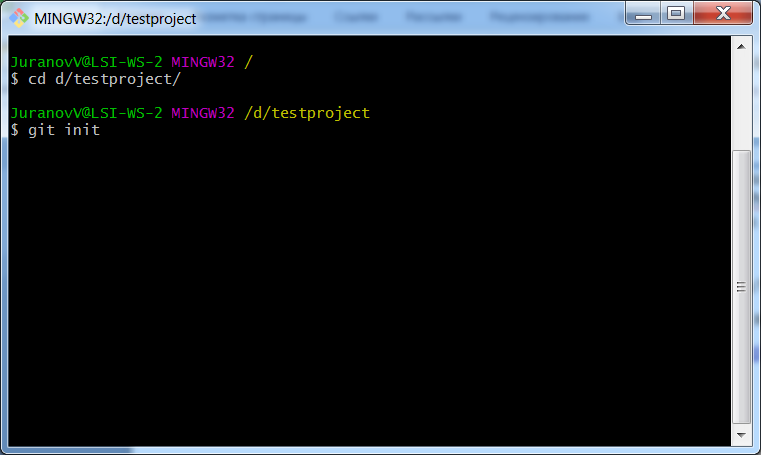


Рисунок 1 – Оболочка Git Bash

Теперь определим папку для проекта и инициализируем в ней новый репозиторий командой git init. Для этого в командной строке с помощью команды cd установим требуемый каталог в качестве текущего, например:

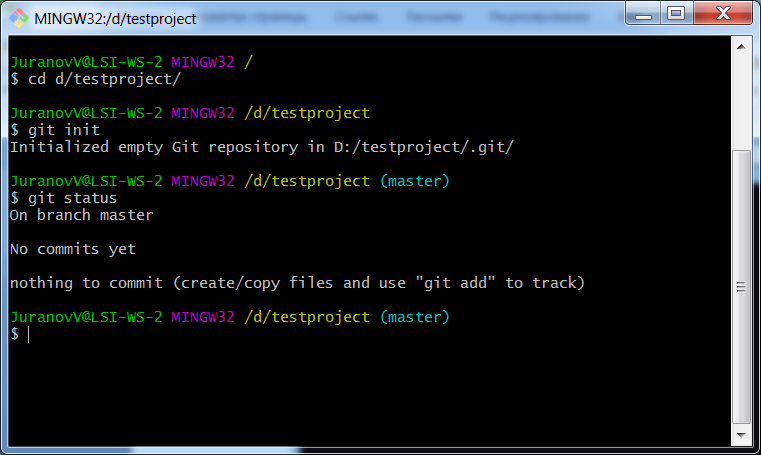


Для создания нового репозитария достаточно просто зайти в папку проекта и набрать git init:

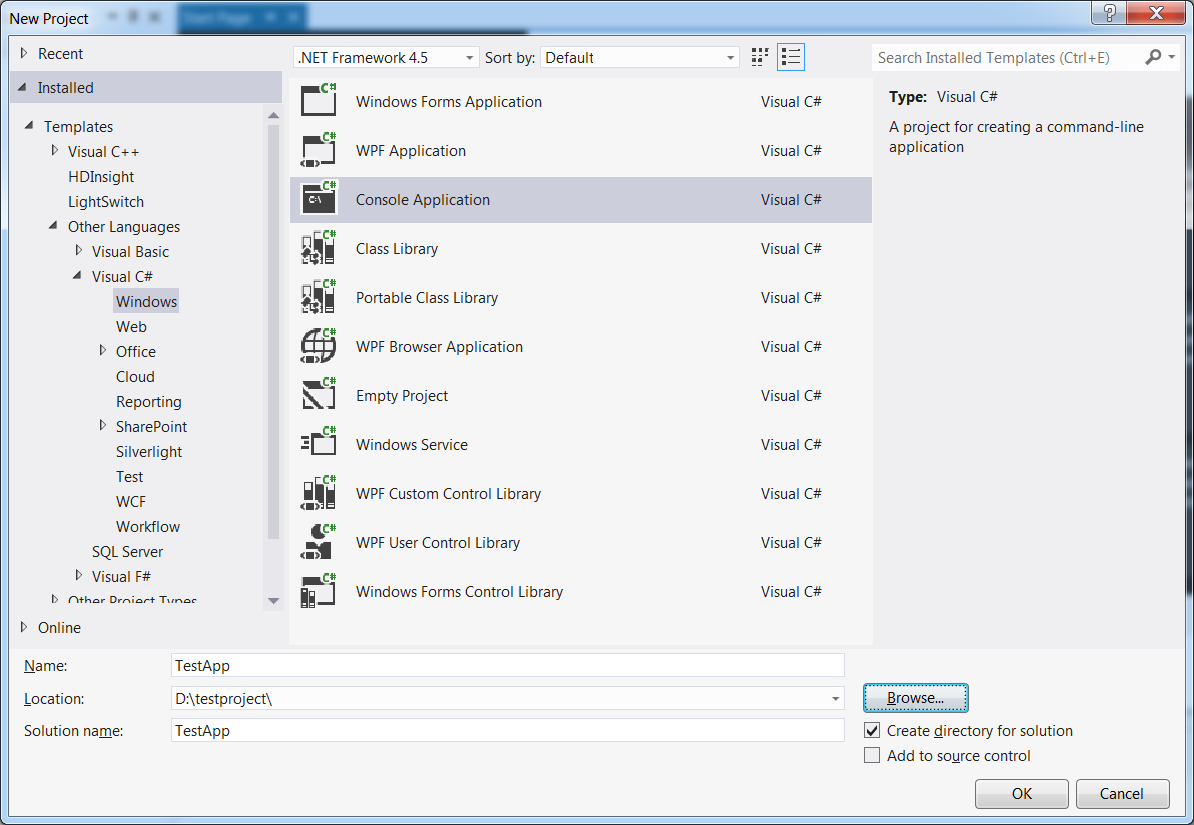


Был создан пустой репозитарий — папка .git в корне проекта, в которой и будет собираться вся информация о дальнейшей работе.

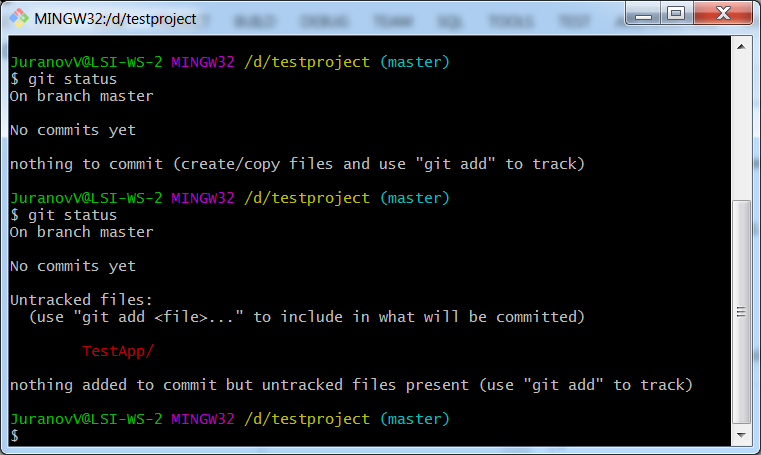
Информацию о статусе нашего репозитория можно проверить командой git status



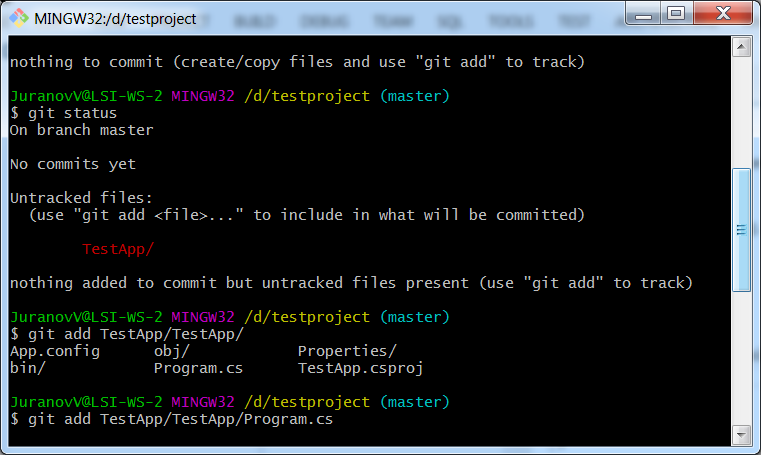
Система сообщила нам о том, что никаких изменений для сохранения в нашей папке нет. Теперь добавим в наш репозиторий проект консольного приложения C# из Visual Studio. Откроем Visual Studio и создадим новый проект, при этом в качестве пути сохранения проекта выберем папку с git репозиторием:



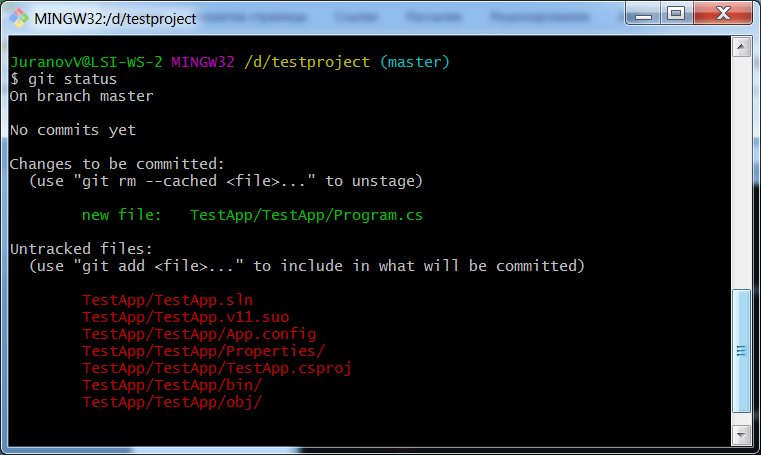
Если теперь мы проверим статус нашего репозитория, то будет выведено сообщение о неотслеживаемых файлах:



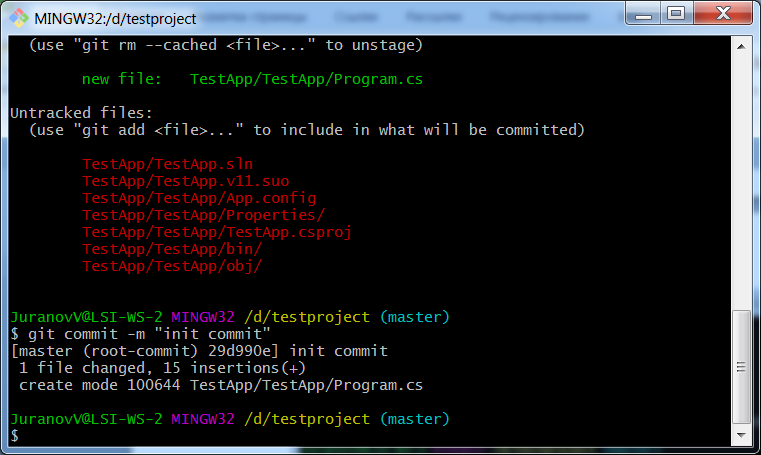
Файлы для отслеживания их состояния через git можно добавлять(индексировать) как по одному, так и сразу все вместе. Пока попробуем проиндексировать один файл. Это будет основной файл с исходным кодом программы по умолчанию – Program.cs. Что бы его проиндексировать, необходимо выполнить команду /, где TestApp/TestApp/ это указание пути к файлу относительно текущего каталога.



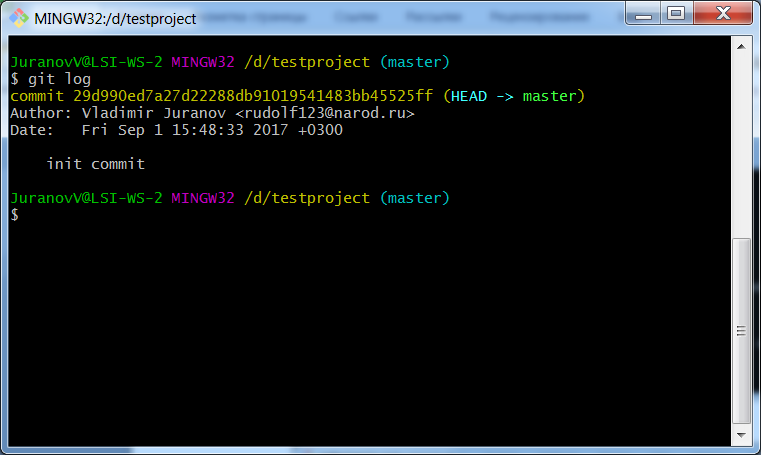
Проверим статус репозитория:



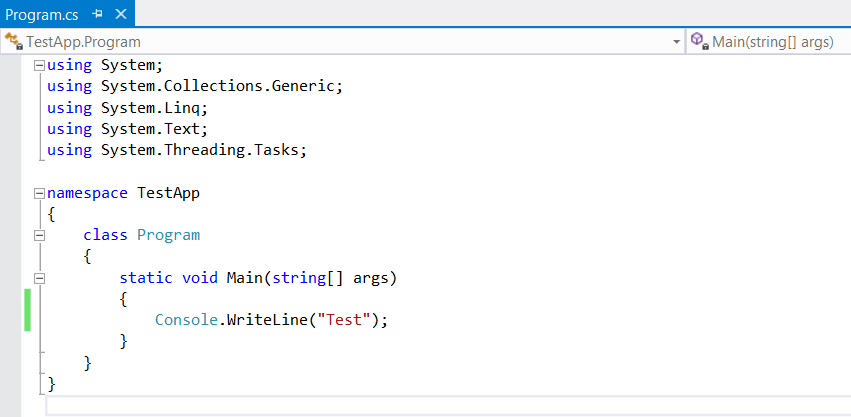
Теперь можем сделать первый коммит в наш репозиторий, то есть мы зафиксируем изменения, которые есть на данный момент в файле Program.cs. Для этого выполним команду git commit -m “init commit”, где «init commit» это просто словесное описание изменений.



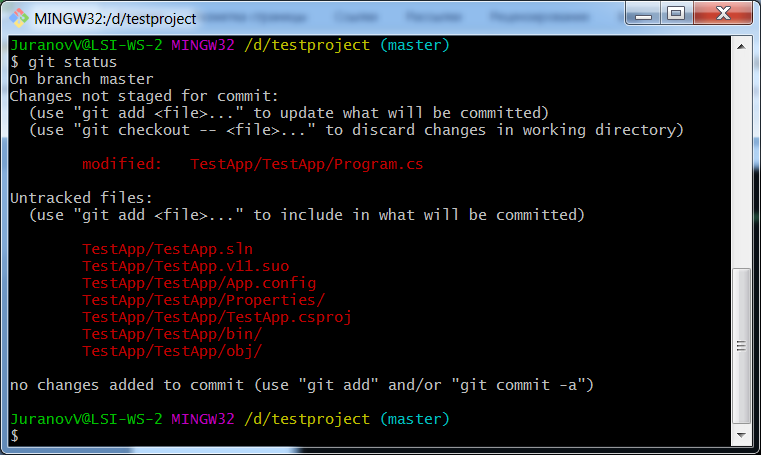
С помощью команды git log мы можем просмотреть всю историю коммитов нашего репозитория:



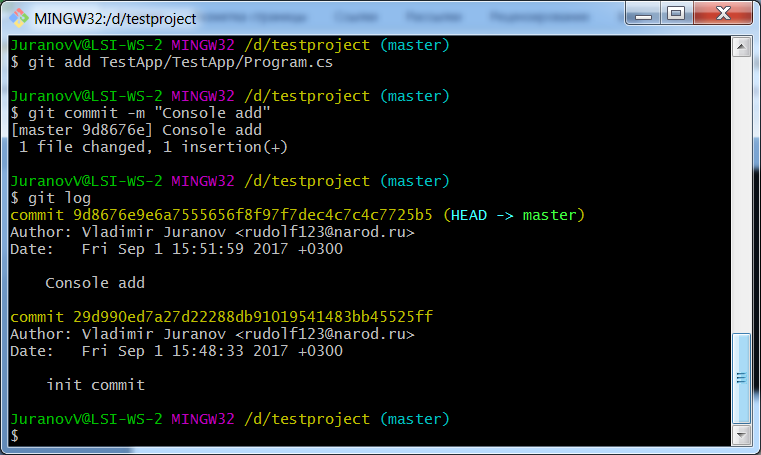
Теперь изменим файл Program.cs и сохраним изменения



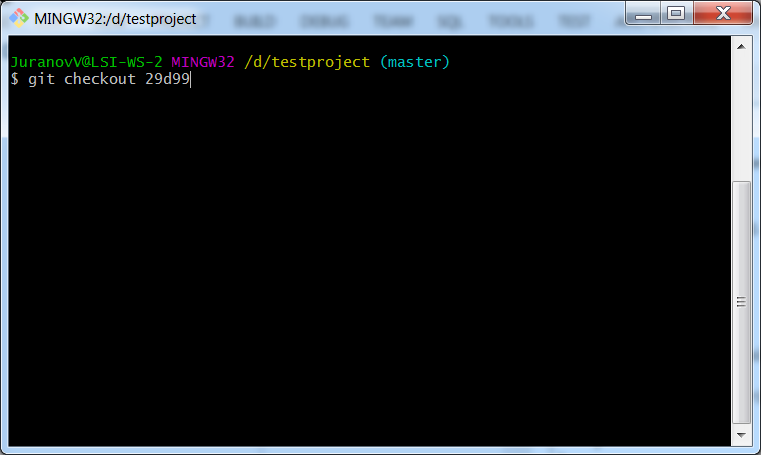
Если проверить статус нашего репозитория, то будет видно, что отслеживаемый файл Program.cs был изменен.



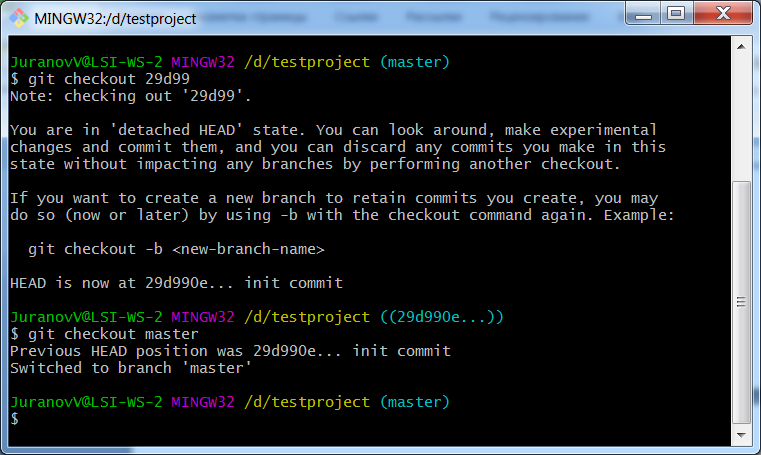
Что бы зафиксировать новые изменения в нашем репозитории опять выполним функцию git add TestApp/TestApp/Program.cs и git commit -m “Console add”. Теперь при просмотре истории репозитория мы видим 2 коммита, каждый из них имеет свой уникальный хеш, по которому мы будем к ним обращаться в дальнейшем.

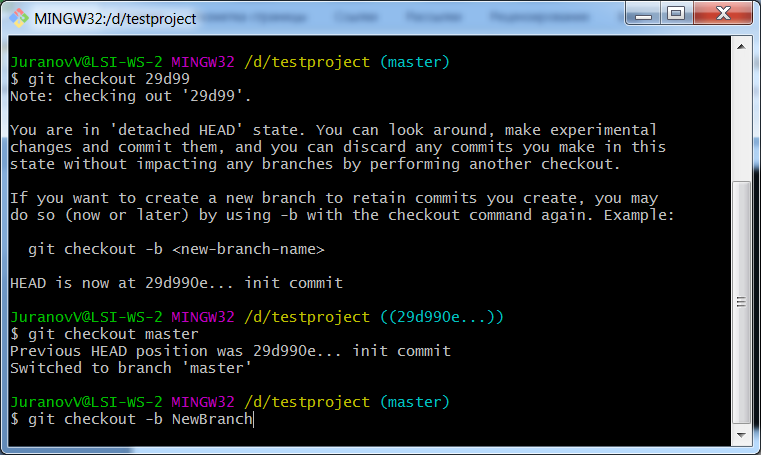


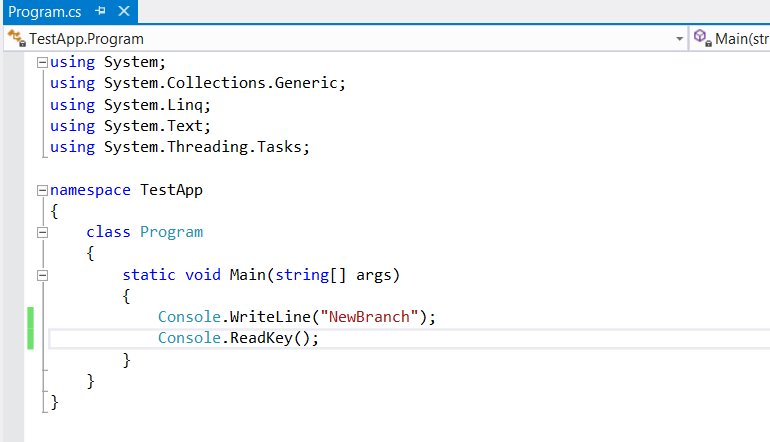
Если в процессе разработки нам надо будет посмотреть как выглядел файл на некотором этапе, то можно просто переключиться на соответствующий коммит командой git checkout 29d99 , где последний параметр это начало хеша коммита(его необязательно переписывать весь). Переключимся на первый коммит.

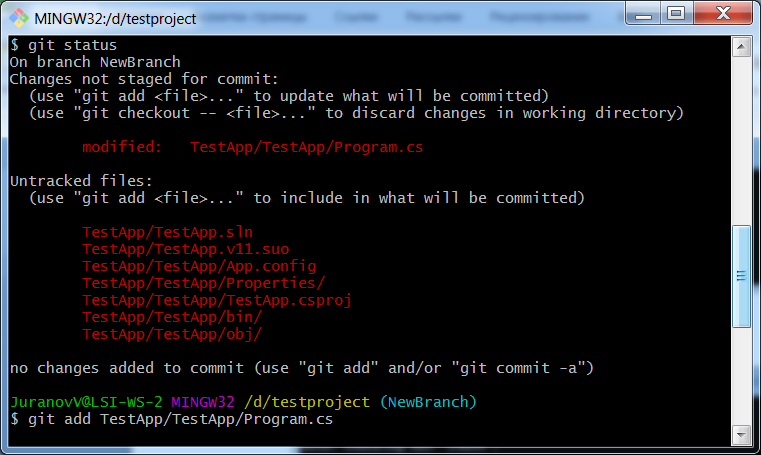


При этом, если мы откроем файл Program.cs, то функция вывода на экран в нем будет отсутствовать, так как она появилась только во втором коммите. Для возврата к последнему коммиту можно набрать git checkout master

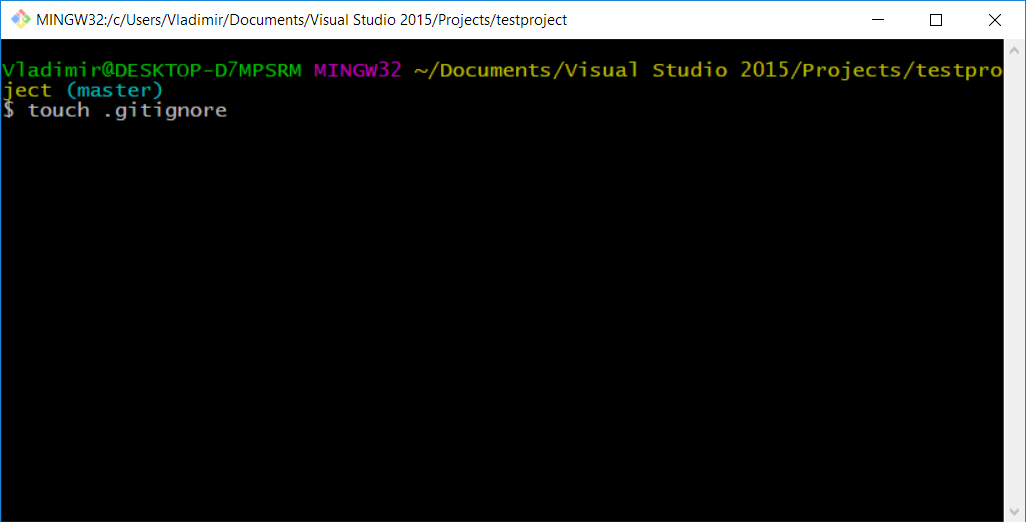








Обычно в систему контроля версий добавляют только те файлы, которые пользователь изменяет самостоятельно, либо которые единожды были созданы средой разработки. Как и многие другие системы Visual Studio автоматически генерирует дополнительные файлы внутри проекта на этапе компиляции программы (например папка obj). Их не принято добавлять в индекс системы Git. Поэтому мы не стали добавлять сразу все файлы проекта в систему Git. Для того чтобы Git игнорировал некоторые файлы или папки используется специальный файл, в котором указываются игнорируемы элементы проекта. Название файла должно быть .gitignore и он должен располагаться в корневой папке репозитория. По умолчанию Windows не позволит создать файл без имени (считается, что после точки идет расширений файла), поэтому можно воспользоваться командной строкой оболочки Git и создать файл командой touch .gitignore



После этого в папке с проектом появится данный файл, открыть его можно любым текстовым редактором. Далее необходимо занести в него директории, изменения в которых мы будем игнорировать:

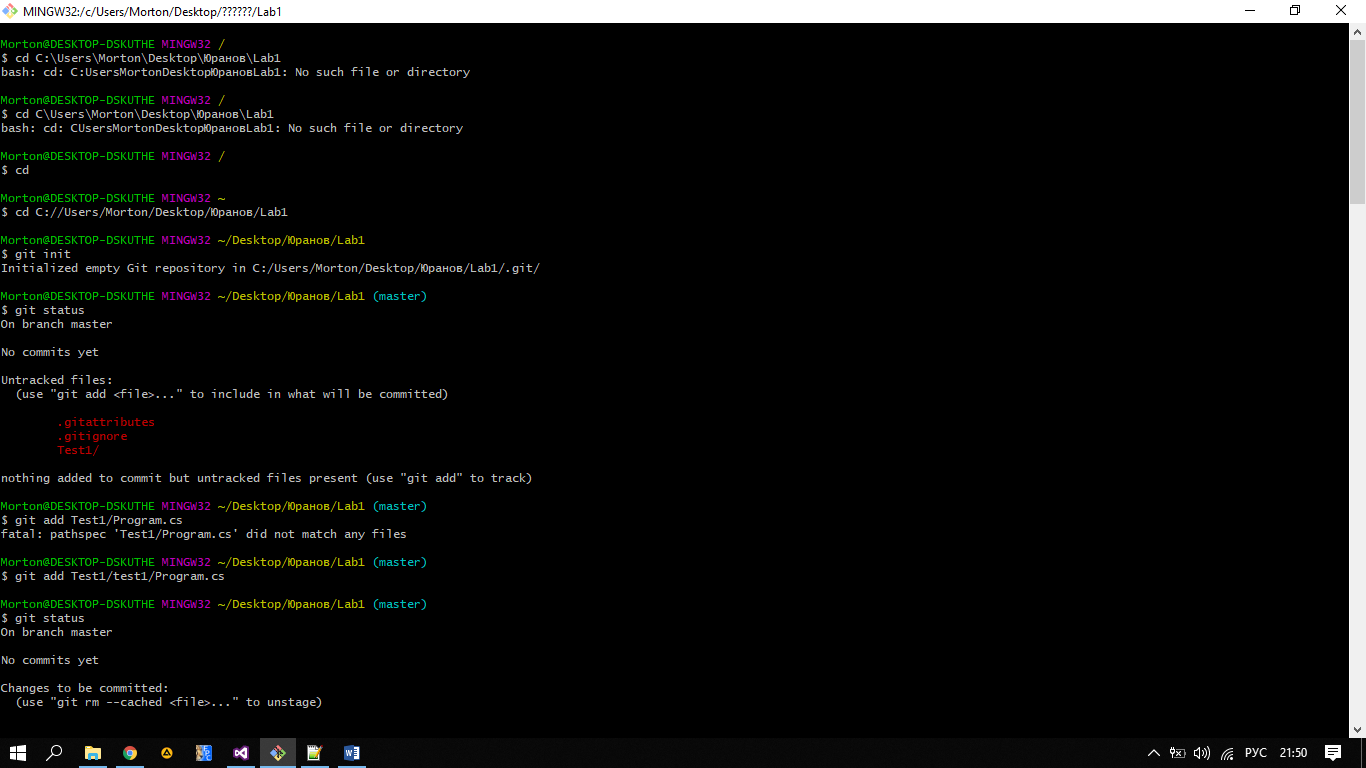
testproject/obj/Debug/

testproject/bin/Debug/

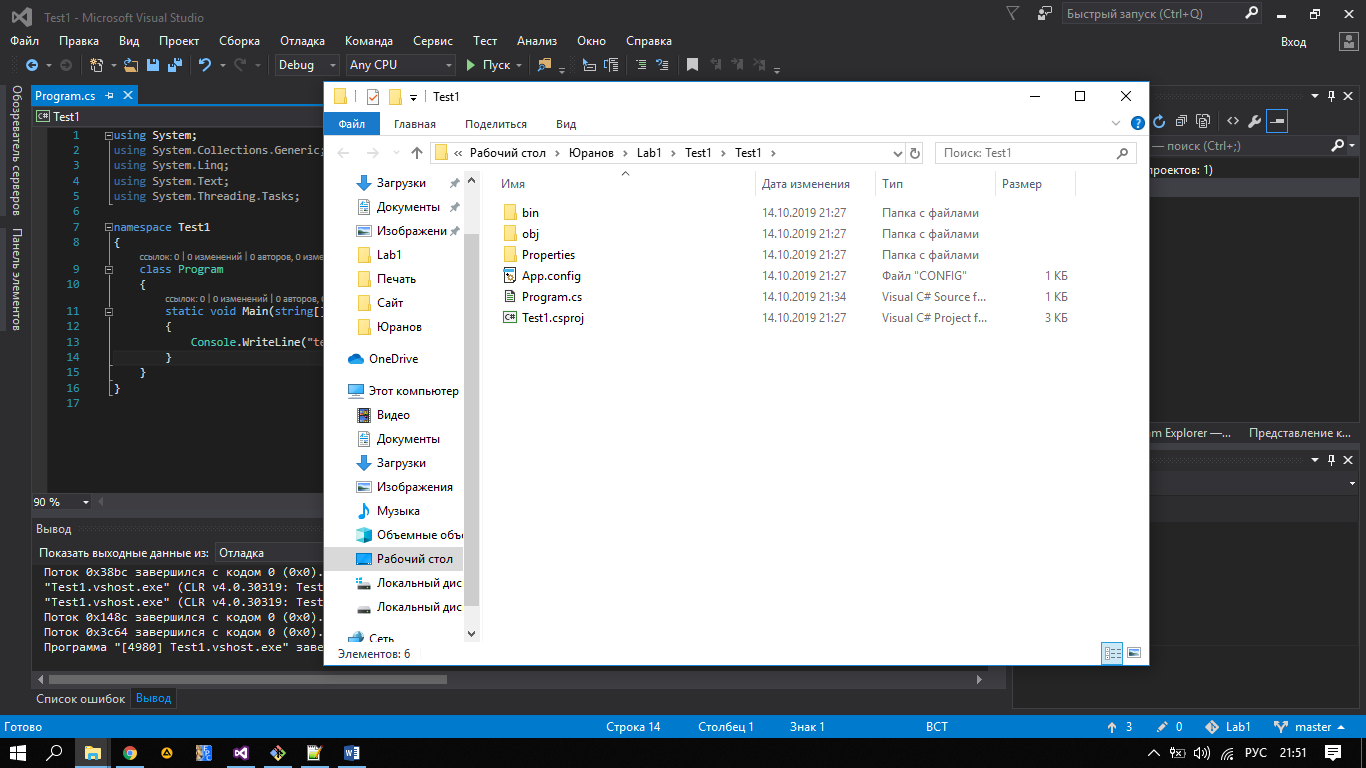
Теперь можно добавить сразу все фалы проекта в систему контроля версий командой git add .

**Задание к лабораторной работе**

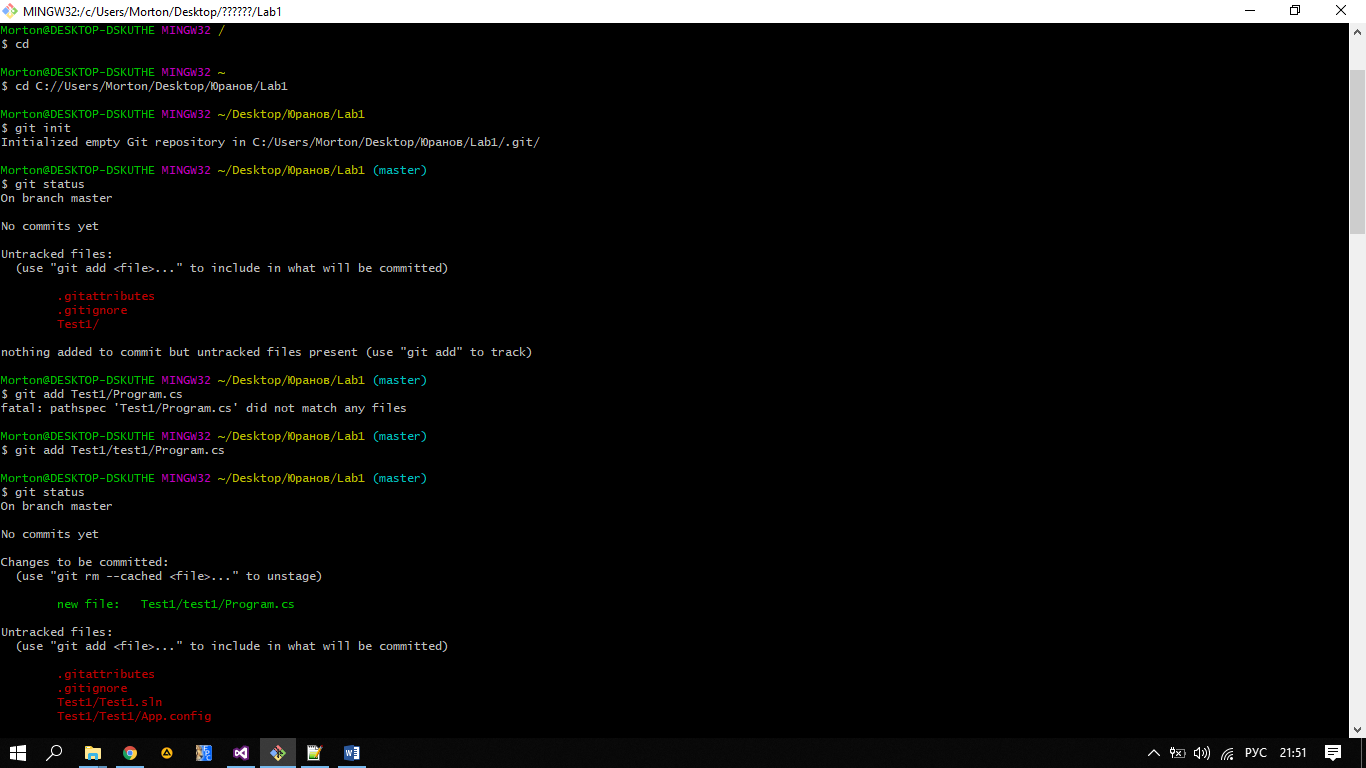
1. Создать репозиторий Git на локальном компьютере.



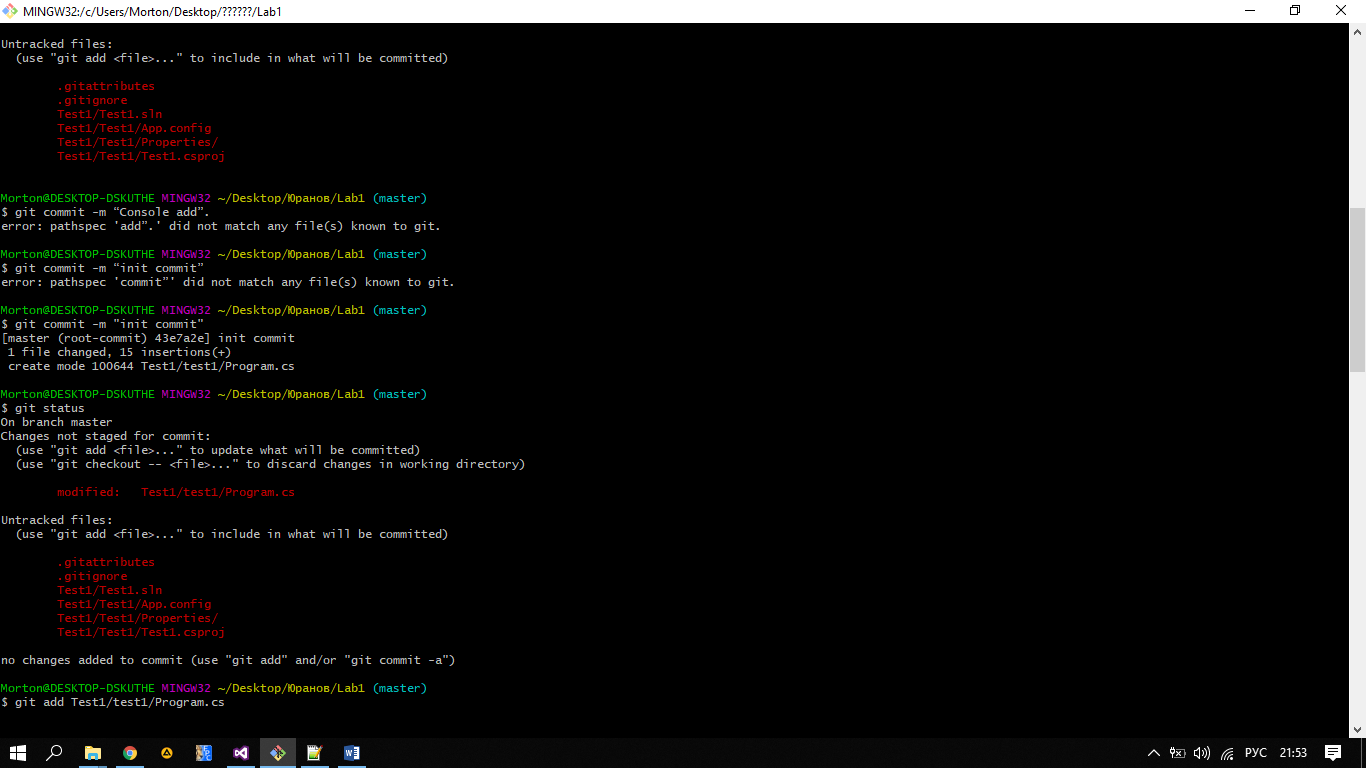
1. Создать проект консольного приложения Visual Studio в папке репозитория.



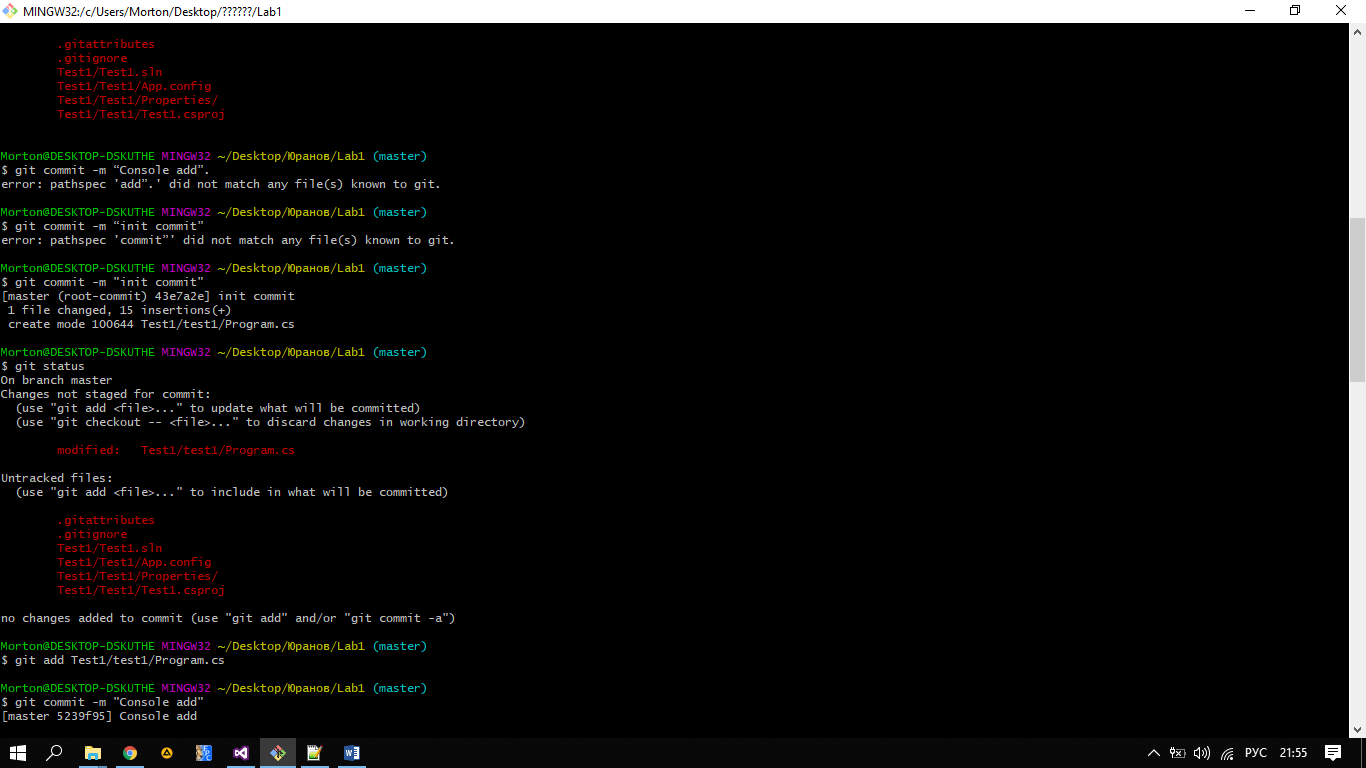
1. Добавить файл Program.cs в систему контроля версий



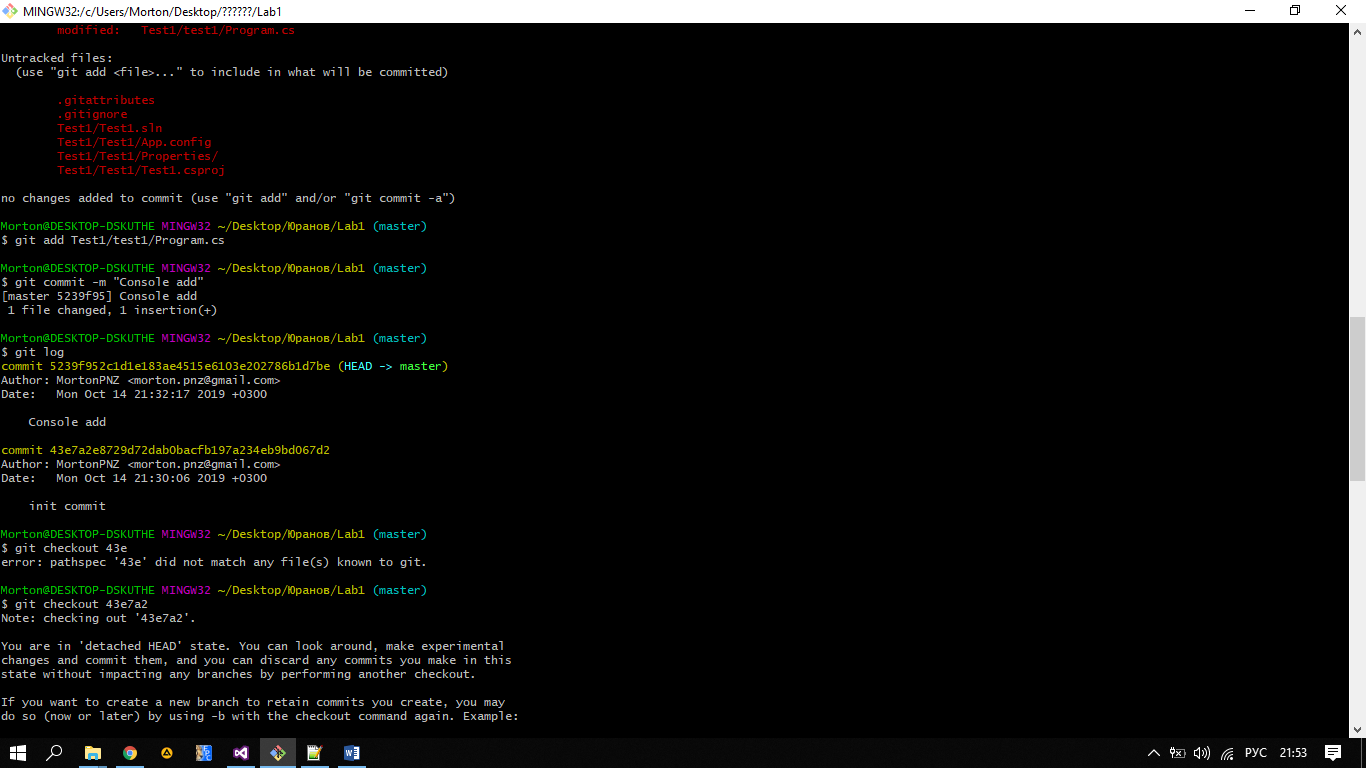
1. Сделать коммит изменений



1. Изменить и сохранить файл Program.cs

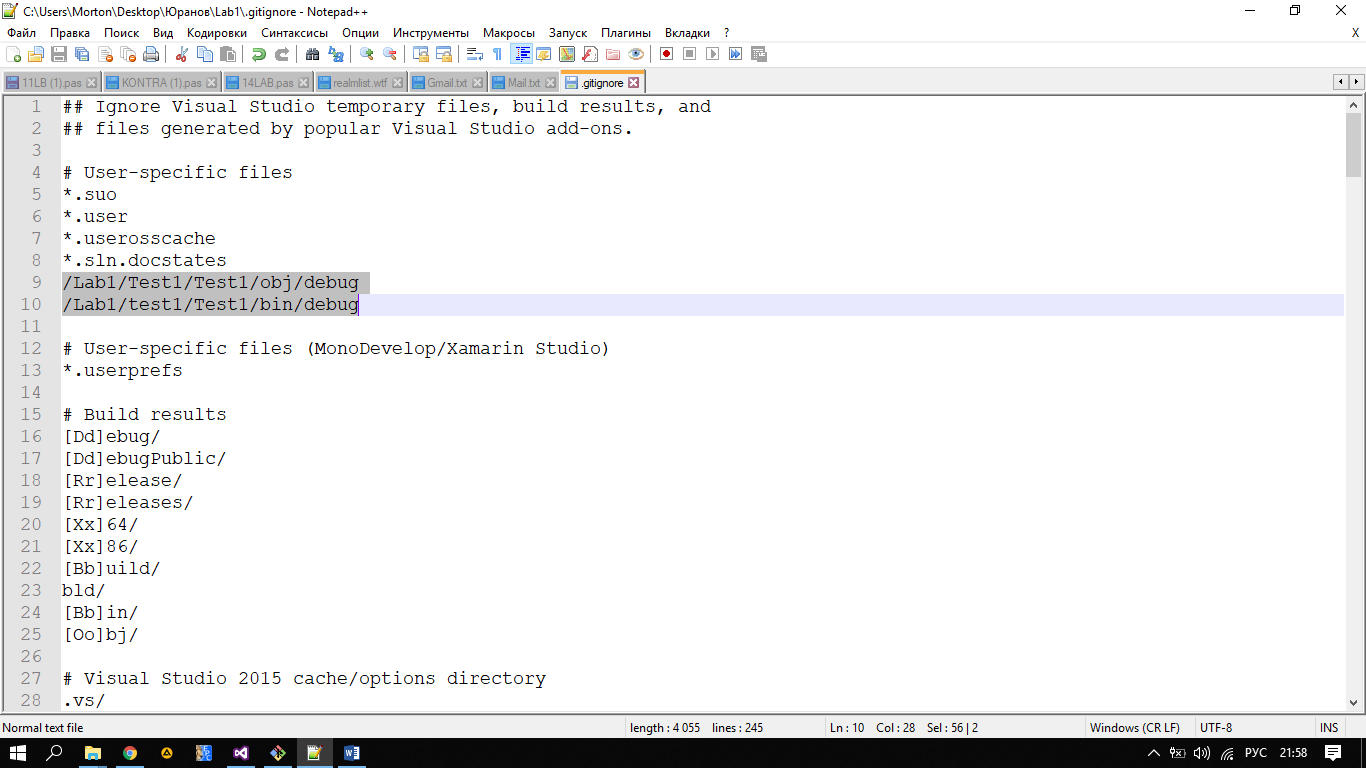


1. Сделать коммит изменений



1. Добавить файл для игнорирования автоматически создаваемых файлов проекта

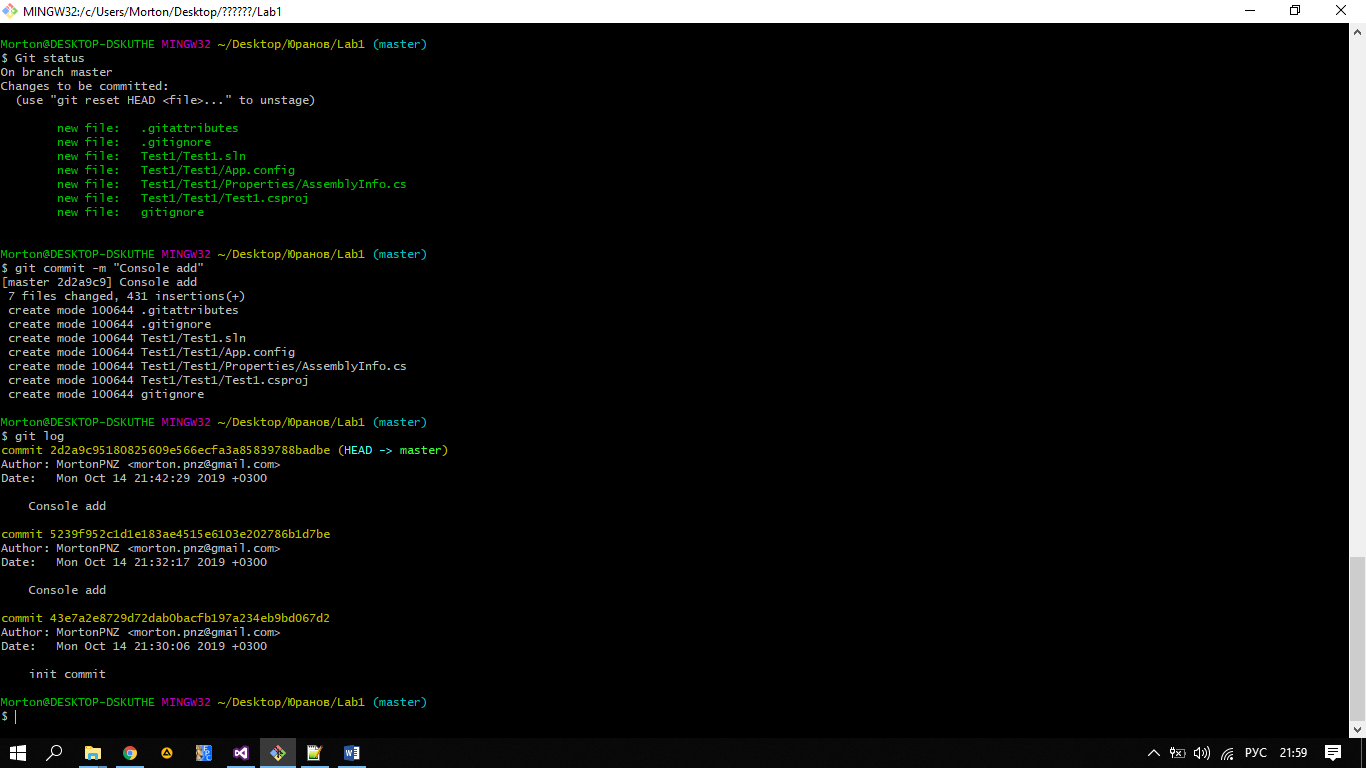




1. Добавить все файлы проекта в систему контроля версий.



1. Сделать коммит изменений



Контрольные вопросы

1. Что такое система контроля версий?

Система контроля версий или система управления версиями - это программа, которая позволяет фиксировать и отслеживать изменения исходного кода программных проектов. При этом, сам программный проект представляет собой дерево каталогов, то есть набор файлов и подкаталогов, хранящихся в каталоге проекта.

1. Что такое Git?

Git — распределённая система управления версиями. По своей сути она создаёт базу изменений файлов, присваивая каждому записанному изменению свой хэш, что позволяет при помощи команд и указанию хэша переместиться к выбранному состоянию файлов

1. Что такое коммит?

Операция позволяющая зафиксировать состояние файлов и придаёт этому состоянию собственных хэш, записывая их в лог

1. Что означает добавить файл в индекс?

Добавление файла в индекс позволяет получать данные о конкретном файле или папке для системы и записывать их в коммит.

1. Как перейти к какому-либо коммиту?

Используя команду «git log» мы можем получить список сохранённых коммитов и переместиться к ним при помощи команды «git checkout \*хэш коммита\*».

1. Как указать системе контроля версий Git какие файлы необходимо игнорировать при индексации?

Для того чтобы Git игнорировал некоторые файлы или папки используется специальный файл, в котором указываются игнорируемы элементы проекта. В файл можно записать как название конкретного файла, так и маску файла. Название файла должно быть .gitignore и он должен располагаться в корневой папке репозитория. По умолчанию Windows не позволит создать файл без имени (считается, что после точки идет расширений файла), поэтому можно воспользоваться командной строкой оболочки Git и создать файл командой touch .gitignore

1. Как добавить все файлы и папки в систему контроля версий?

Добавить сразу все фалы проекта в систему контроля версий командой «git add .».