**Национальный технический университет Украины**

**«Киевский политехнический институт имени Игоря Сикорского»**

**Факультет информатики и вычислительной техники**

**Кафедра автоматики и управления в технических системах**

**Отчет по лабораторной работе №1**

**«Эмпирические методы программной инженерии»**

Выполнил:

Студент группы ЗПИ-ЗП 61

Макиян С. А.

Проверила:

ассистент

Хмелюк М. С.

Киев 2019

**Лабораторная работа №1. Часть 1.**

**1. Тема**

Системы контроля версий. Распределенная система контроля версий Mercurial.

**2. Задание**

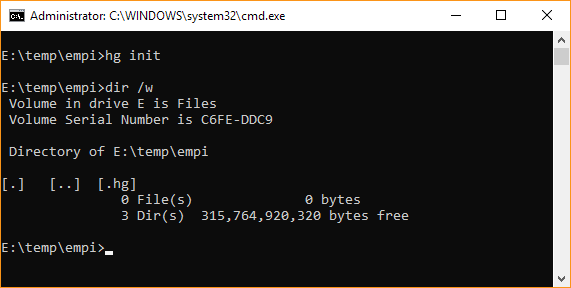
Представление системы контроля версий Mercurial и отличия от других систем контроля версий.

Создать репозиторий в добавить в него несколько файлов с помощью командной строки и графической оболочки TortoiseHg.

Работа с командами: создание репозитория, добавление файлов, сохраниние файлов в репозиторий, просмотр истории изменений, зафиксированных в репозитории, возвращение измененных файлов к зафиксированному в репозитории виду, отображение списка измененных файлов, изменение содержимого файла команда, удаление файлов из репозитория, отображение содержимого любого файла для любой ревизии, команда апдейта.

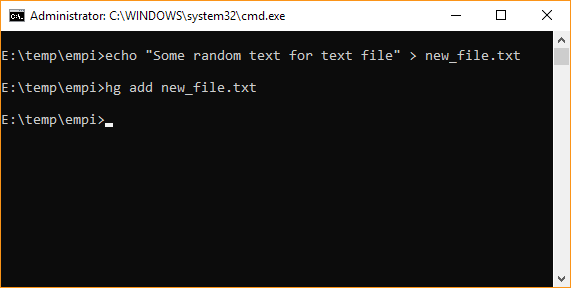
**3.1 Создание Mercurial репозитория с помощью командной строки.**

Команда *hg init*



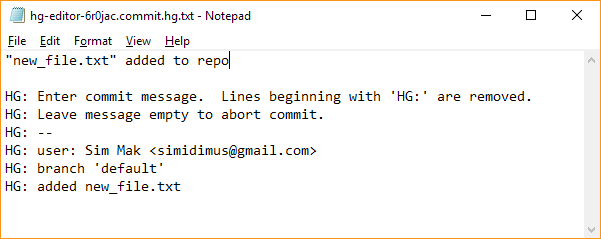
**3.2 Добавление новых файлов в репозиторий.**

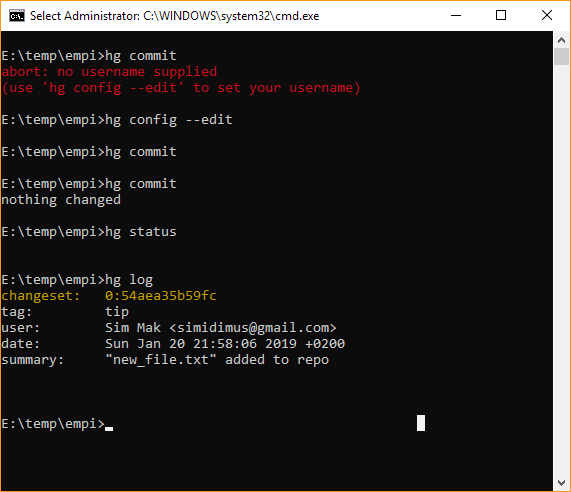
Команда *hg add*



**3.3 Сохранение файлов в репозитории.**

Команда *hg commit.* Для первого раза, hg попросит создать пользователя, под которым будет это изменение сохранено в репозиторий. Вводим имя и e-mail, сохраняем файл. После этого повторно вводим команду *hg commit* и заполняем сообщение. Проверяем результат с помощь команды *hg log.*





**Вывод:**

Создание пустого репозитория и добавление в него файла с соответствующим комментарием произошло успешно.

**Лабораторная работа №1. Часть 2.**

**1. Тема**

Работа с ветками. Shelving в системе контроля версий Mercurial.

**2. Задание**

Научиться создавать ветки, работать в разных ветках (вносить изменения), осуществлять переходы между веткам.

Для проекта создать 3 ветки. В каждой из них реализовать разный функционал (например, 1 ветка вывести сообщение, 2 ветка – вывести другое сообщение, 3 ветка – по нажатию на кнопку нарисовать квадрат).

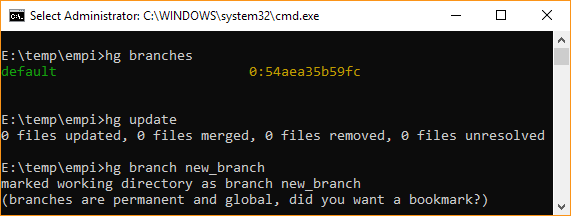
Работа с командами commit, update.

Научиться работать с функцией Shelve (откладывание изменений в системе контроля версий).

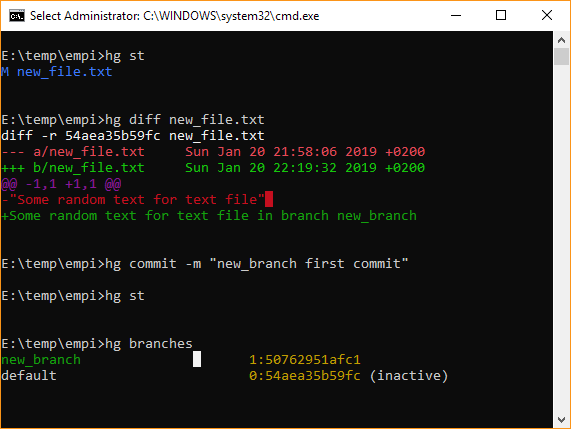
Добавить файл, внести в него изменения, отложить на полочку, достать с полочки.

**3.1 Создание новой ветки**

Команда *hg branch.*

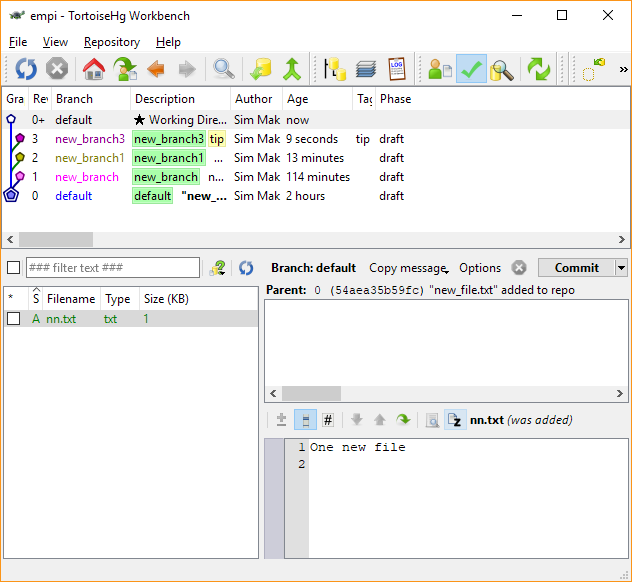
****

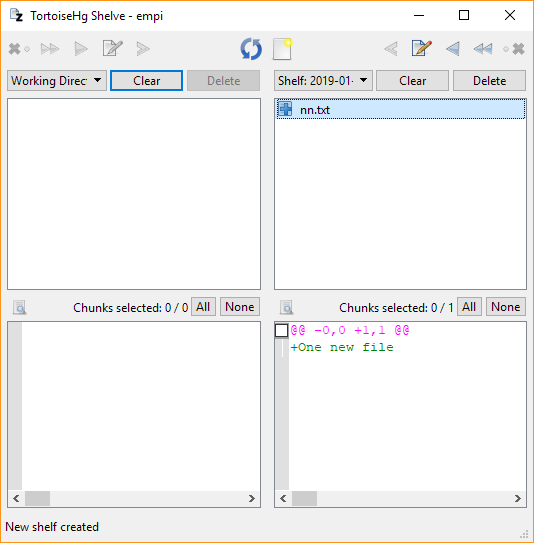
Внесем изменения в файл, и сохраним их.

****

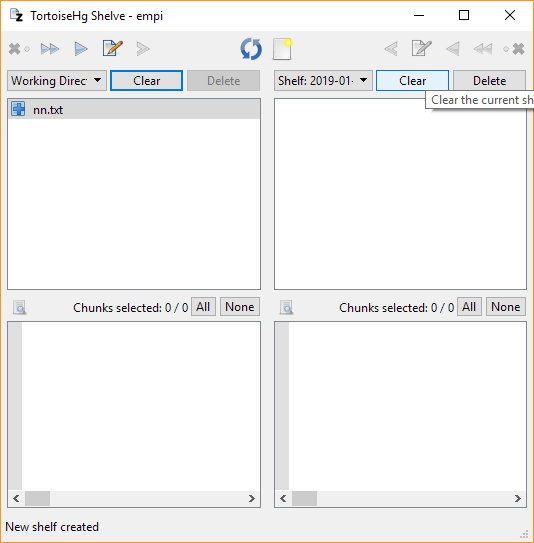
Повторим тоже самое еще два раза, создав ветки new\_branch2, new\_branch3.

Перейдем на ветку default, создадим новый файл, добавим его, и отложим на полку.



****

А затем вернем файл с полки.

****

**Национальный технический университет Украины**

**«Киевский политехнический институт имени Игоря Сикорского»**

**Факультет информатики и вычислительной техники**

**Кафедра автоматики и управления в технических системах**

**Отчет по лабораторной работе №2**

**«Эмпирические методы программной инженерии»**

Выполнил:

Студент группы ЗПИ-ЗП 61

Макиян С. А.

Проверила:

ассистент

Хмелюк М. С.

Киев 2019

**Лабораторная работа №2**

**1. Тема**

Работа со слиянием изменений в системе контроля версий Mercurial.

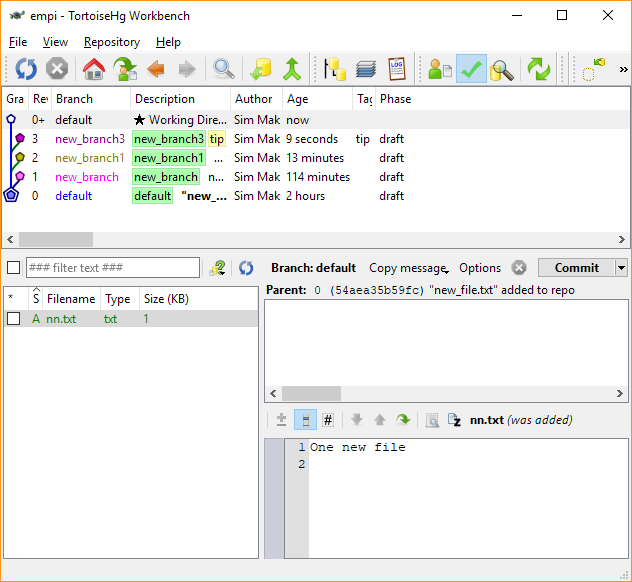
**2. Задание**

Создать несколько веток. В разных ветках проекта внести изменения таким образом, чтобы создать конфликтную ситуацию при слиянии (в разных ветках в файле с одинаковым именем с разным содержимым).

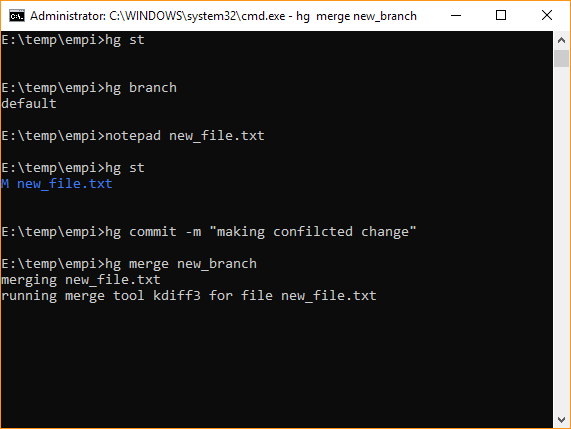
Слить 2 ветки, разрешив конфликты вручную используя KDiff3.

**3.1 Слить две ветки с конфликтными файлам.**

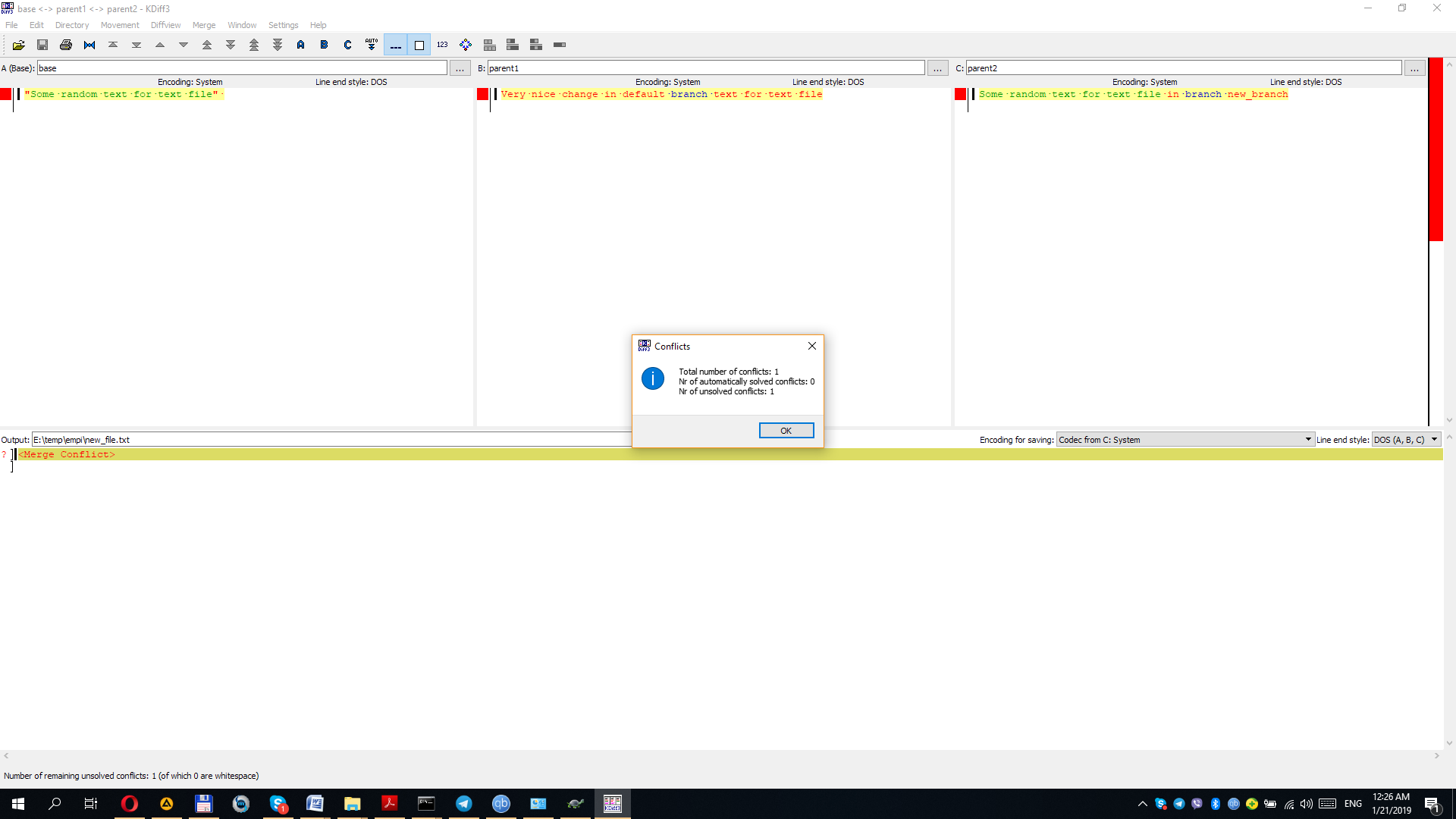
Отобразим текущее состояние веток



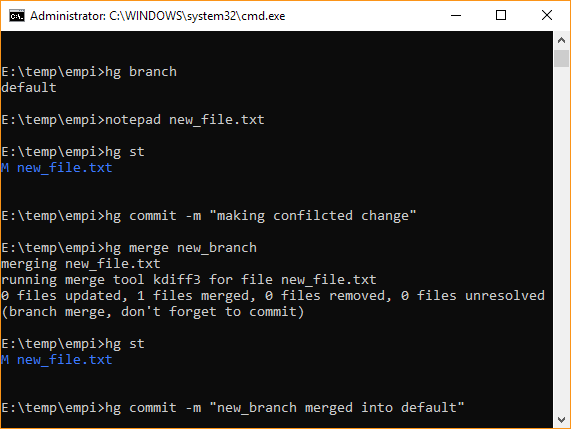
Переключимся на ветку default



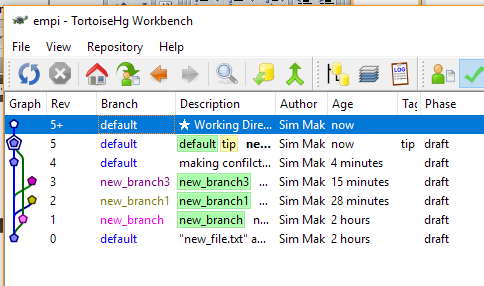
Запустим мерж с конфликтной ситуацией.



Сохраним изменения



Проверим графы в графической оболочке



**Вывод.**

В репозитории были созданы несколько веток, а так же конфликтная ситуация. Все при слиянии веток, конфликты были устранены через визуальную утилиту для мержа.

**Национальный технический университет Украины**

**«Киевский политехнический институт имени Игоря Сикорского»**

**Факультет информатики и вычислительной техники**

**Кафедра автоматики и управления в технических системах**

**Отчет по лабораторной работе №3**

**«Эмпирические методы программной инженерии»**

Выполнил:

Студент группы ЗПИ-ЗП 61

Макиян С. А.

Проверила:

ассистент

Хмелюк М. С.

Киев 2019

**Лабораторная работа №3**

**1. Тема**

Mercurial. Центральный репозиторий. Клоны. Робота разных пользователей с репозиторием.

**2. Задание**

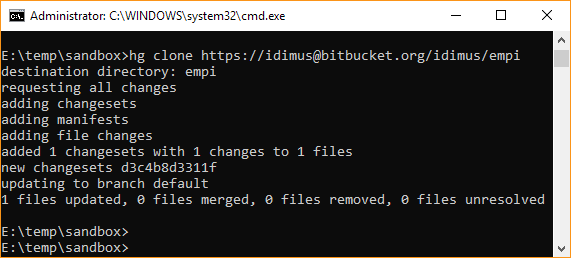
Научиться работать в команде. Создать центральный репозиторий на https://bitbucket.org/ или на локальной машине. Клонировать 2 репозитория на локальной машине для работы «двух пользователей» (два репозитория создать под одним пользователем). Добавить проект в одном репозитории, сделать commit, протолкнуть в центральный репозиторий (push), зайти в другой репозиторий и забрать проект с центрального репозитория (pull). В двух репозиториях сделать изменения в проекте, добавить изменения в разные ветки запушить из одного репозитория, получить в другой репозиторий решив конфликты слияния.

**3.1 Создадим репозиторий на bitbucket**

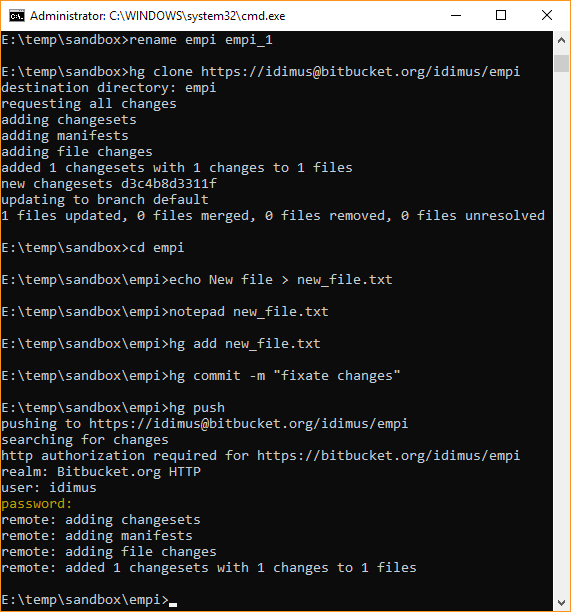
Создадим нового пользователям и репозитория на bitbucket.

Адрес репозитория <https://idimus@bitbucket.org/idimus/empi>.

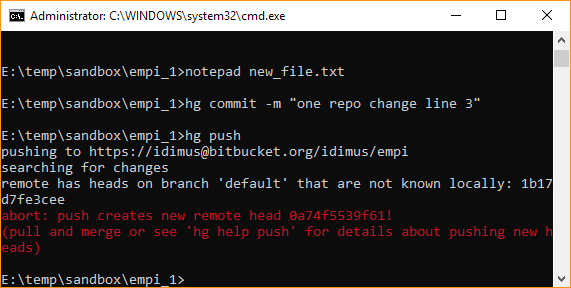
**3.2 Сделаем две локальных копии репозиториев**

****

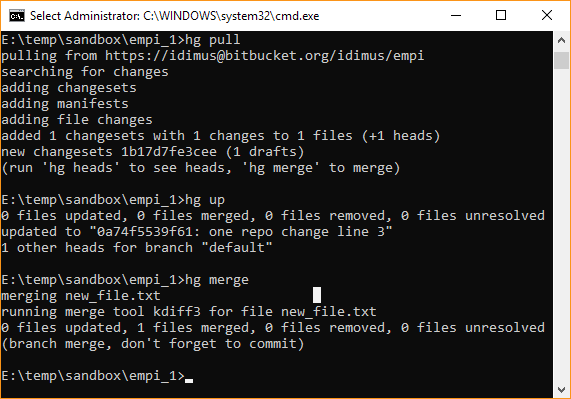
**3.3 Создадим конфликтную ситуацию**

****

Делаем изменения в одинаковых местах из двух репозиториев, пытаемся залиться со второго, получаем ошибку.



Делаем merge



**Вывод.**

Был создан удаленным репозиторий, и две локальных его копии. При изменениях в одних и тех же местах, возникла конфликтная ситуация, которая разрешилась командой мерж.

**Национальный технический университет Украины**

**«Киевский политехнический институт имени Игоря Сикорского»**

**Факультет информатики и вычислительной техники**

**Кафедра автоматики и управления в технических системах**

**Отчет по лабораторной работе №4**

**«Эмпирические методы программной инженерии»**

Выполнил:

Студент группы ЗПИ-ЗП 61

Макиян С. А.

Проверила:

ассистент

Хмелюк М. С.

Киев 2019

**Лабораторная работа №4**

**1. Тема**

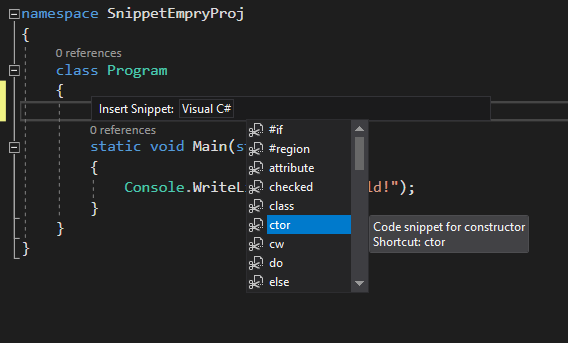
Фрагменты кода (snippets) В Visual Studio.

**2. Задание**

Познакомиться с фрагментами кода и окружением кода фрагментами (Фрагменты типа Surround With) и их использованием. Научиться изменять существующие и создавать свои фрагменты кода.

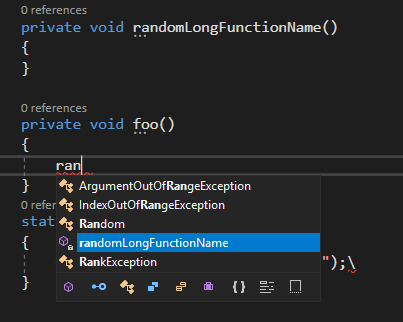
**3.1 Вставка готовых снипетов**

Нажимае Ctrl+K, Ctrl+X

****

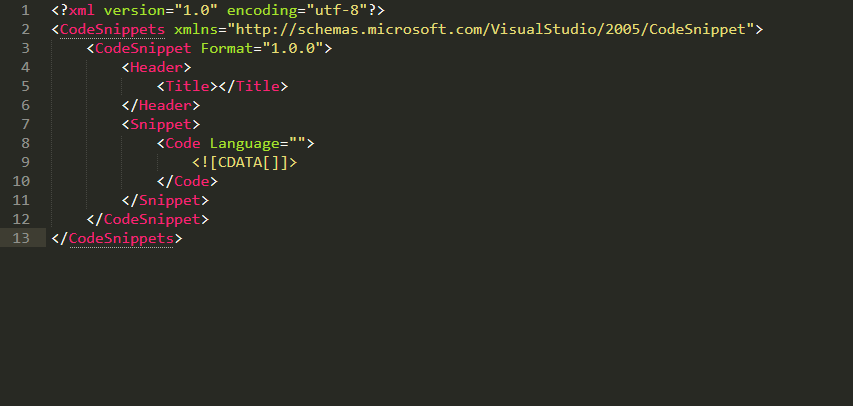
**3.2** **Использование фрагмента кода с помощью элемента shortcut фрагмента кода**

Начинаем набирать нужным нам тест, нажиаме Ctrl+Space

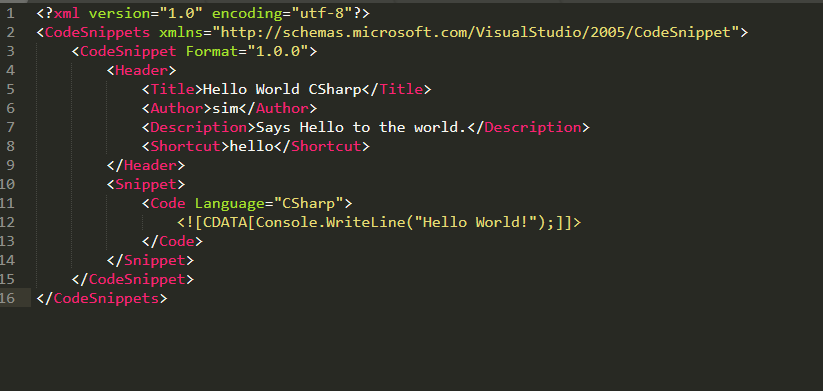


**3.3 Создание собственных фрагментов кода**

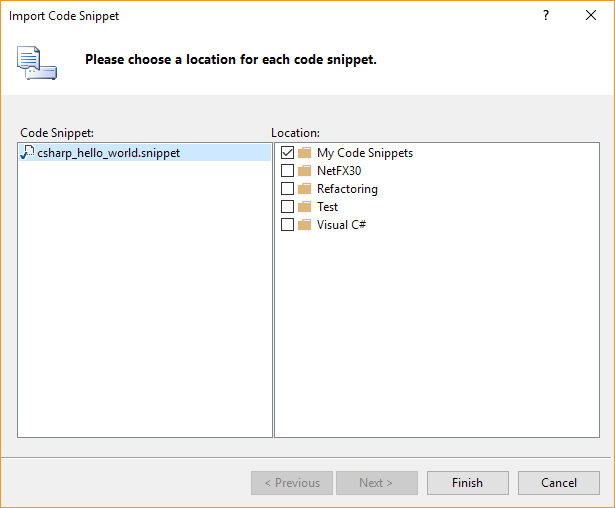
Возьмем темплейт для снипета и сохраним его в файл, с расширением snippet.



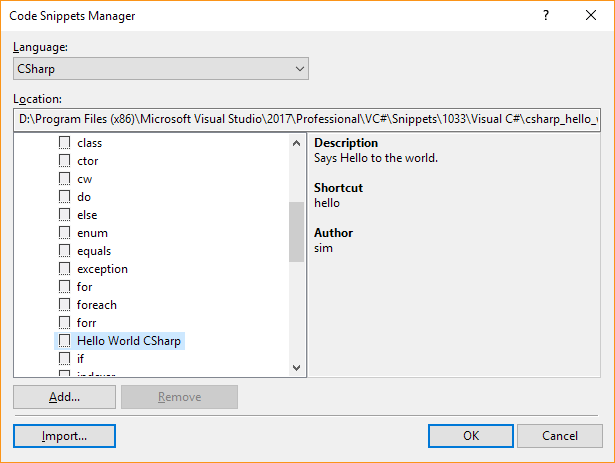
Допишем в сниппет нужную нам информацию.



Запустим Snippet Manager и импортируем снипет.



Проверим что снипет добавился



Вставим снипет в код. Снипет вставился.

**Вывод:**

Было рассмотрено что такое снипеты и как ими пользоваться. Так же в ходе лабораторной работы был создан свой снипет и успешно применен.

**Национальный технический университет Украины**

**«Киевский политехнический институт имени Игоря Сикорского»**

**Факультет информатики и вычислительной техники**

**Кафедра автоматики и управления в технических системах**

**Отчет по лабораторной работе №5**

**«Эмпирические методы программной инженерии»**

Выполнил:

Студент группы ЗПИ-ЗП 61

Макиян С. А.

Проверила:

ассистент

Хмелюк М. С.

Киев 2019

**Лабораторная работа № 5**

**1. Тема**

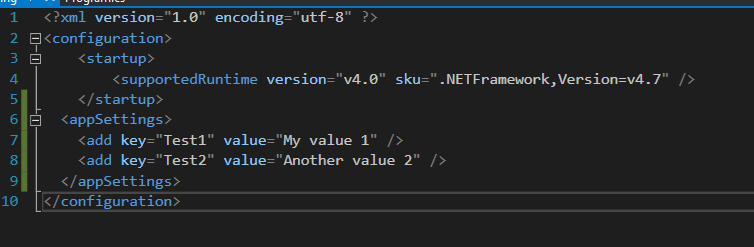
Файлы конфигурации приложения.

**2. Задание**

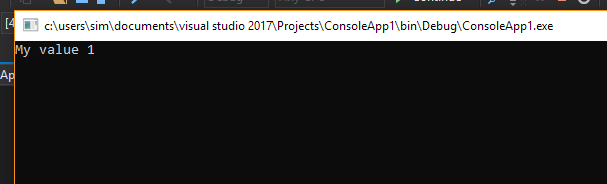
Разобраться с файлами конфигурации приложения. Научиться считывать и записывать данные в файл конфигурации App.config.

**3.1 Добавить вручную ключи в файл конфигурации**

Откроем App.config и добавим ключи вручную.

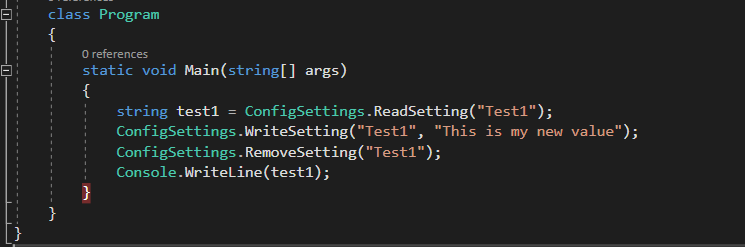


Считаем их и распечатаем в консоль.



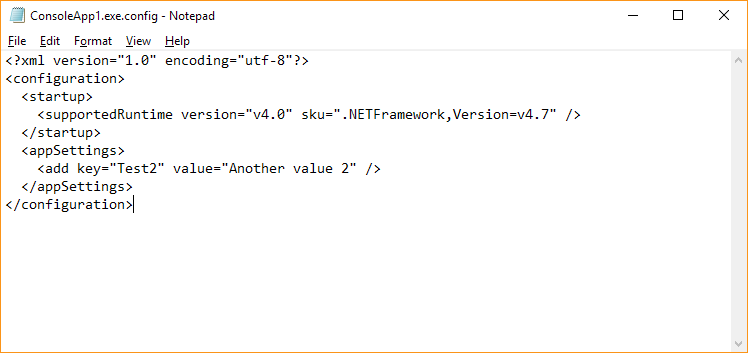
**3.2 Запись и редактирование файла конфигурации**

Добавим код класса ConfigSettings. Используем его



После повторного запуска программы увидим, что больше в конколи нет никаких надписей, как при первом запуске.

Откроем файл конфигурации и убедимся, что больше нет ключа Test1



**Вывод.**

В ходе лабораторной работы, были успешно добавлены и отредактированы ключи в файле конфигурации (App.Config), которые после компиляции программы, хранятся в ConsoleApp1.exe.config. С помощью данного механизма, можно сохранять и редактировать настройки и данные программы.

**Национальный технический университет Украины**

**«Киевский политехнический институт имени Игоря Сикорского»**

**Факультет информатики и вычислительной техники**

**Кафедра автоматики и управления в технических системах**

**Отчет по лабораторной работе №6**

**«Эмпирические методы программной инженерии»**

Выполнил:

Студент группы ЗПИ-ЗП 61

Макиян С. А.

Проверила:

ассистент

Хмелюк М. С.

Киев 2019

**Лабораторная работа №6**

**1. Тема**

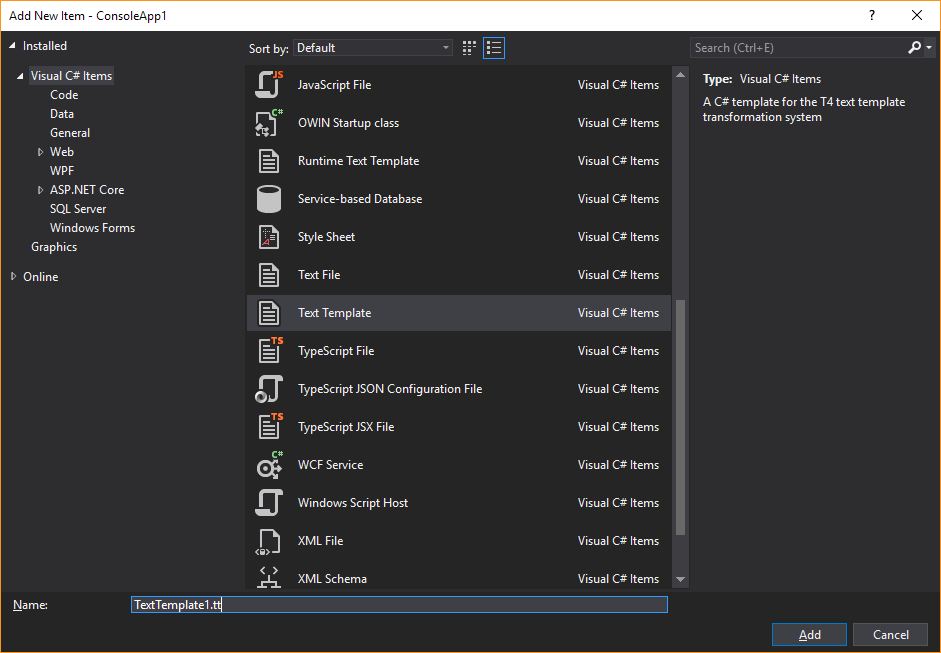
Шаблоны Т4 в Visual Studio.

**2. Задание**

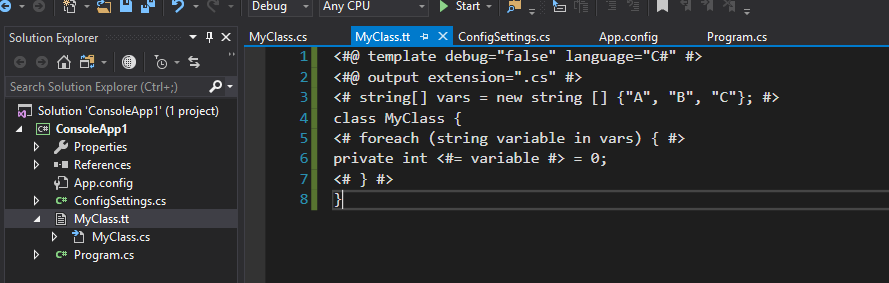
Разобраться с созданием и использованием шаблонов Т4.

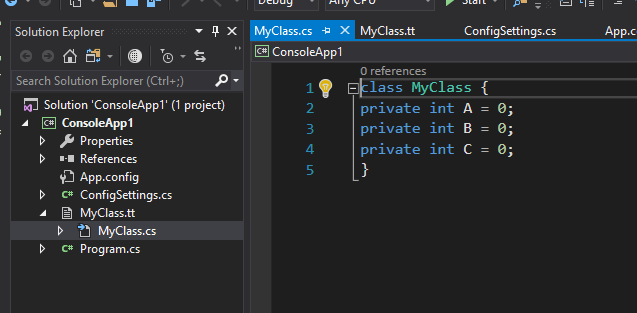
**3.1 Выполнение**

Создадим новый файл темплейта

****

Поместим туда пустой шаблонный код, как пример



Просмотрим результат, работы этого темплейта.  


Добавим нехватающих секций в темплейт

<#@ template debug="false" language="C#" #>

<#@ output extension=".cs" #>

<# string[] vars = new string [] {"A", "B", "C"}; #>

class MyClass {

<# foreach (string variable in vars) { #>

private int <#= variable #> = 0;

<# } #>

<# for (int i=0;i < 2;i++) { #>

public void <#= CreateFunctionName(i)#>()

{

}

<# } #>

}

<#+ public string CreateFunctionName(int counter)

{

string result = "";

switch (counter)

{

case 0:

result = "Foo";

break;

case 1:

result = "Bar";

break;

}

return result;

} #>

Результат работы темплейта.

class MyClass {

private int A = 0;

private int B = 0;

private int C = 0;

public void Foo()

{

}

public void Bar()

{

}

}

**Вывод**

Технология T4 (Text Templating Transformation Toolkit) является мощным инструментом для генерации код на основе шаблонов. С его помощью, можно создавать скелеты будущего однообразного кода и экономить время, а так же избежать ошибок, свойственных процедуре Copy/Paste.