1. **Пример работы с NUnit.**  
     
   NUnit - открытая среда юнит-тестирования приложений для .NET, которая позволяет создавать автоматические тесты. Данный вид тестов обладает рядом преимуществ:

* Высокое качество программы;
* Снижение стоимости;
* Безопасность регрессии сети

Чем выше качество программы, тем меньше средств затрачивается на устранение недостатков проекта. То есть, если найти недостатки в проекте на раннем этапе, решить их будет дешевле.  
  
**Создание проекта библиотечного класса**

Теперь создадим простой класс теста, используя C#, который добавляет два показателя, и протестируем его. Чтобы создать тест класса в Visual Studio, используя С#, выполните следующие шаги:

**Шаг 1**: Создание проекта библиотечного класса

Откройте Visual Studio.

Для создания проекта библиотечного класса сначала нажмите File -> New -> Project, выберите Visual C # в установленном шаблоне, выберите Class Library. Введите подходящее название (мы используем IntroNUnitTest), нажмите "OK".

После того, как Вы кликнули кнопку "OK", Вы увидите Ваш проект в Solution Explorer (в правом верхнем углу Visual Studio).

**Шаг 2**: Добавьте ссылки к Вашему проекту.

Чтобы добавить ссылку NUnit к Вашему проекту, кликните правой кнопкой мыши на ссылку, выберите Add reference-> Browse, затем nunit.framework.dll и «OK».

**Шаг 3**: Создание тестового класса.

Кликните правой кнопкой мыши на Вашем проекте, выберите Add -> Class, введите имя (используем MyNUnitTestClass) и далее "Add".

После того, как Вы кликнули на кнопку Add, Вы увидите класс.

После того, как Вы создали класс, добавьте using NUnit.Framework.

Следует соблюдать некоторые условия и понятия:

* Каждый класс должен содержать атрибут [TestFixure] и должен быть общедоступен.
* В каждом методе должен присутствовать атрибут [Test].
* Оператор подтверждения об отсутствии ошибок: Булевские значения, описывающие, что должно быть ключевым словом, когда выполняется действие.
* Ожидаемое исключение: один из типов исключения, который мы ожидаем во время выполнения тест-метода.
* Установка: программа, которая запускается перед выполнением каждого тест-метода (например, регистрация в системе конкретного пользователя или инициализация одноэлементных классов).
* Демонтаж: программа запускается после окончания каждого тест-метода (например, удаление строк из таблицы, которые были вставлены во время теста).

На примере мы постараемся сравнить две переменные х и у. Каждой переменной присвоили значение 20. В общей сложности написали 4 теста (PositiveTest, NegativeTest, ExpectedExceptionTest и NotImplementedException).

### **Запуск тестов**

После написания всех тестов в тестовом классе необходимо запустить тесты, чтобы проверить, проходят ли они успешно. Для запуска тестов перейдите в папку NUnit, выберите NUnit Application (.exe) и дважды кликнике по ней, выберите команду File-> Open Project, выберите проект (в нашем случае это IntroNUnitTest.dll), затем кликаем Run.

Далее Вы увидите количество тестов, которые не сработали, и количество невыполненных тестов.  
Если Вы хотите запустить какой-то определенный тест, кликните правой кнопкой мыши в тесте, который Вы хотите запустить, и выберите Run test(s).

1. **Пример работы с хUnit.**  
     
    хUnit - это собирательное название семейства [фреймворков](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) для [модульного тестирования](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), структура и функциональность которых основана на [SUnit](https://ru.wikipedia.org/w/index.php?title=SUnit&action=edit&redlink=1), предназначавшегося для [языка программирования](https://ru.wikipedia.org/wiki/%D0%AF%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [Smalltalk](https://ru.wikipedia.org/wiki/Smalltalk). *SUnit*, разработанный [Кентом Беком](https://ru.wikipedia.org/wiki/%D0%91%D0%B5%D0%BA,_%D0%9A%D0%B5%D0%BD%D1%82) в 1998 году, был написан в [высоко структурном](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) [объектно-ориентированном](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) стиле, получил широкую популярность и был адаптирован для множества других языков. Названия фреймворков этого семейства образованы аналогично "SUnit", обычно заменяется буква "S" на первую букву (или несколько первых) в названии предполагаемого языка ("[JUnit](https://ru.wikipedia.org/wiki/JUnit)" для [Java](https://ru.wikipedia.org/wiki/Java), "[NUnit](https://ru.wikipedia.org/wiki/NUnit)" для программной платформы [.NET](https://ru.wikipedia.org/wiki/.NET_Framework) и т. д.). Семейство таких фреймворков с общей [архитектурой](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F) обычно и известно как "xUnit".  
     
   **Архитектура xUnit**

Все фреймворки из семейства xUnit имеют следующие базовые компоненты архитектуры, которые в различных реализациях могут слегка варьироваться.

1. Модуль, выполняющий тестирование (Test runner)

Модуль представляет собой исполняемую программу, которая выполняет тесты, реализованные с помощью фреймворка, и отображает информацию о ходе их выполнения.

1. Тестовые сценарии (Test cases)

Варианты тестирования (тестовые сценарии/случаи) являются базовыми элементами модульных тестов.

1. Конфигурации тестирования (Test fixtures)

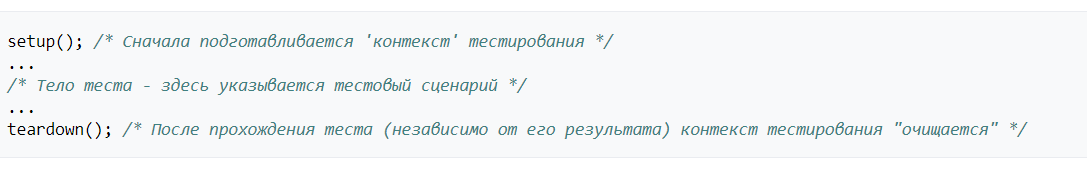
Конфигурация тестирования (также называемая контекстом) — это набор предварительно заданных условий или состояний объектов, необходимый для запуска теста. Разработчик должен задать заведомо корректную конфигурацию перед выполнением каждого теста, а затем вернуть оригинальную конфигурацию после завершения теста.

1. Наборы тестов (Test suites)

Тестовый набор — это несколько тестов, имеющих общую конфигурацию. Очередность выполнения тестов не должна иметь значения.

1. Выполнение тестов (Test execution)

Выполнение каждого теста происходит согласно следующей схеме:



1. Форматирование результатов тестирования (Test result formatter)

Модуль, выполняющий тестирование, должен вывести результаты в одном или нескольких заданных форматах. В дополнение к обычному тексту, воспринимаемому человеком, часто результаты выводятся в формате XML.

1. Утверждения (Assertions)

Утверждение в тесте — это функция или макрос, которая проверяет поведение или состояние тестируемого модуля. Часто утверждением является проверка равенства или неравенства некоторого параметра модуля ожидаемому результату. Неудачное прохождение проверки приводит к провалу всего тестового сценария и (если необходимо) к исключению, которое останавливает сценарий без перехода к следующему утверждению.