Лабораторная работа № 20

Создание Unit-тестов в Visual Studio — C#

Цель:

- Изучить пошаговый процесс создания простейшего Unit-теста в системе Microsoft Visual Studio 2017 (C#) для приложения типа Console App (.NET Framework).
- Научиться создавать собственные Unit-тесты.
- Использовать класс Assert для проведения тестирования работы функций.

План работы:

1. Создать приложение по шаблону Console App (.NET Framework)
2. Подготовка текста модуля Program.cs
   2.1. Добавить функцию Min() в текст модуля
   2.2. Сделать класс Program общедоступным (public)
3. Текст программы, которую нужно протестировать
4. Создание теста
   4.1. Добавление нового проекта к решению
   4.2. Структура решения
   4.3. Текст файла «UnitTest1.cs». Атрибуты [TestMethod] и [TestClass]
   4.4. Выполнение изменений в тексте модуля UnitTest1.cs. Изменение названия метода, который будет тестировать
   4.5. Подключение проекта MinApp к проекту TestMinApp
   4.6. Внесение изменений в текст модуля UnitTest1.cs
      4.6.1. Добавление пространства имен MinApp в модуле UnitTest1.cs
      4.6.2. Текст метода TestMin()
   4.7. Текст модуля UnitTest1.cs
5. Запуск теста на выполнение и проверка результата тестирования
6. Итог. Взаимодействие между проектами

Условие задачи

Для приложения типа Console App (.NET Framework) разработать Unit-тест, который тестирует работу функции Min(). Для функции Min() установить метод тестирования TestMin(). Проверить работу функции.

Выполнение

1. Создать приложение по шаблону Console App (.NET Framework)

Запустить на выполнение Visual Studio. Чтобы создать проект по шаблону Console App (.NET Framework) нужно вызвать следующую последовательность команд
File -> New -> Project...
В результате откроется окно New Project. В окне выбрать шаблон Console App (.NET Framework) как изображено на рисунке 1. Шаблон выбирается во вкладке
Visual C# -> Windows Desktop -> Console App (.NET Framework)

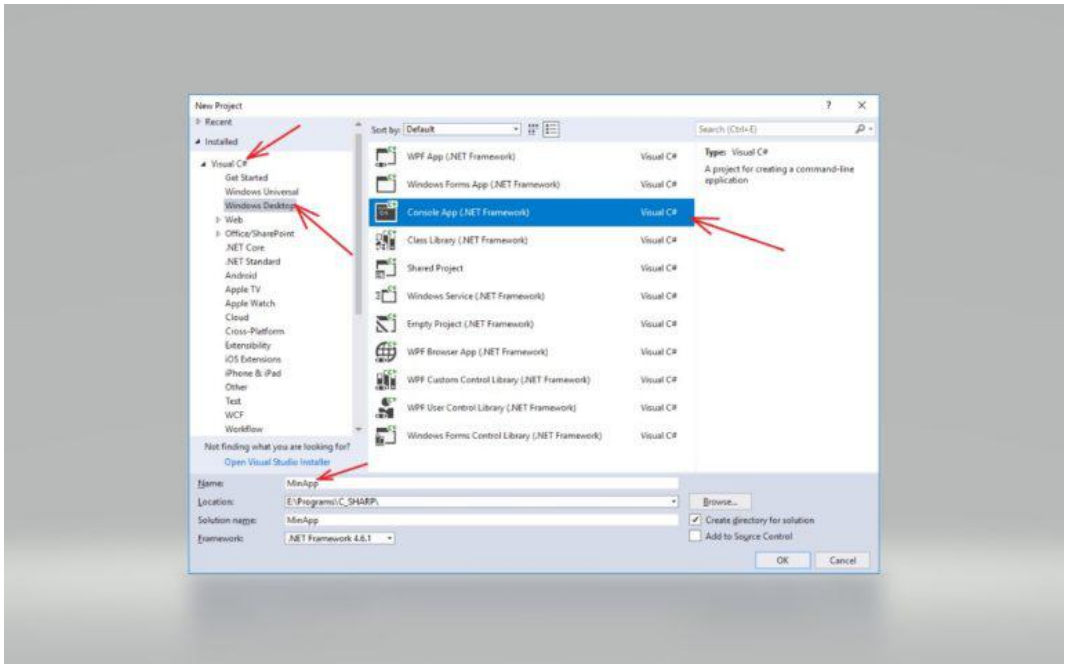

Рис. 1. Окно «New Project». Выбор приложения типа Console App (.NET Framework)

2. Подготовка текста модуля Program.cs

2.1. Добавить функцию Min() в текст модуля

В тело класса Program нужно добавить текст функции Min().
Функция объявляется как общедоступная (public) и статическая (static). Текст функции Min()

```
public static int Min(int a, int b, int c)
{
    int min = a;
    if (min > b) min = b;
    if (min > c) min = c;
    return min;
}
```

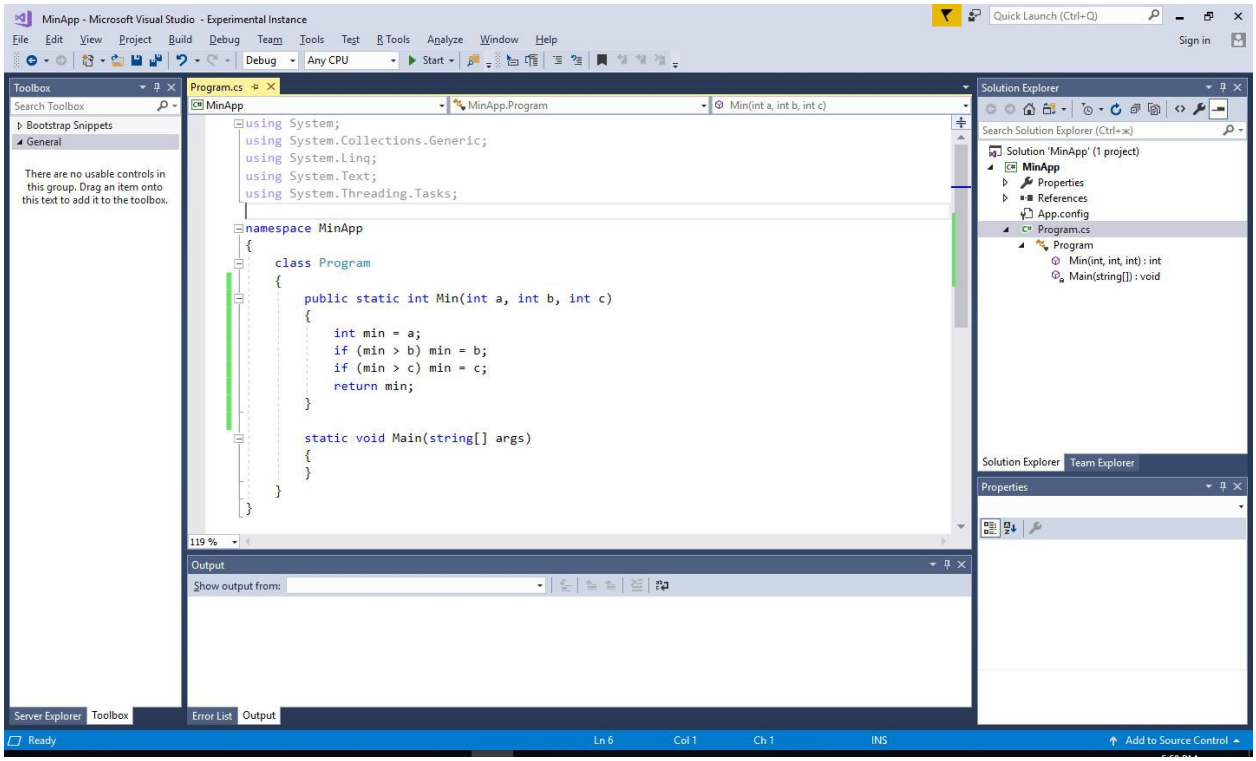

Рис. 2. Вид окна MS Visual Studio 2017, модуль «Program.cs»

2.2. Сделать класс Program общедоступным (public)

Для того, чтобы иметь доступ к функции Min() класса Program, нужно сделать этот класс общедоступным. Для этого, перед объявлением класса, нужно установить ключевое слово public.

```
...
namespace MinApp
{
    public class Program
    {
        // методы класса
        // ...
    }
}

...
```

После этого программа которую нужно протестировать, готова к тестированию.

3. Текст программы, которую нужно протестировать

На данный момент текст программы, которую нужно протестировать, имеет вид:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace MinApp
{
    public class Program
    {
        public static int Min(int a, int b, int c)
        {
            int min = a;
            if (min > b) min = b;
            if (min > c) min = c;
            return min;
        }

        static void Main(string[] args)
        {
            Console.WriteLine("Demo of Unit-testing in C#");
        }
    }
}
```

Поскольку, эта программа будет тестироваться из другого модуля тестирования, то в функции Main() больше ничего вводить не нужно. Так как, в соответствии с условием задачи, нужно протестировать работу функции Min(). А это уже будет осуществляться из модуля тестирования. На данный момент наша программа готова к тестированию.

4. Создание теста

Тест создается отдельным проектом (Project) в решении (Solution). Программа, которая будет тестироваться не знает об этом. Программа-тест, которая будет тестировать вызывает функции программы, которая тестируется. В нашем случае программа-тест будет вызывать функцию

```
int Min(int, int, int);
```

4.1. Добавление нового проекта к решению

Для данного решения (Solution) добавляется новый проект с помощью команды

`File->Add->New Project...`

Окно создания нового проекта изображено на рисунке 3.

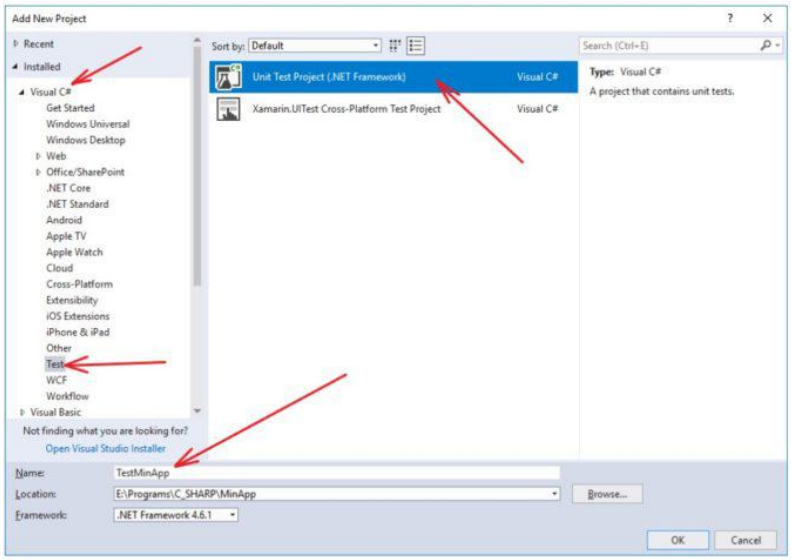

Рис. 3. Окно создания проекта типа Test Project

В окне выбирается группа шаблонов Visual C# -> Test. Из отображенных шаблонов выбирается шаблон проекта «Unit Test Project (.NET Framework)». В поле «Name» указывается имя проекта, который будет тестировать нашу программу. Нужно задать, например, TestMinApp. Проект размещается в отдельной папке «H:\Test\MinApp».

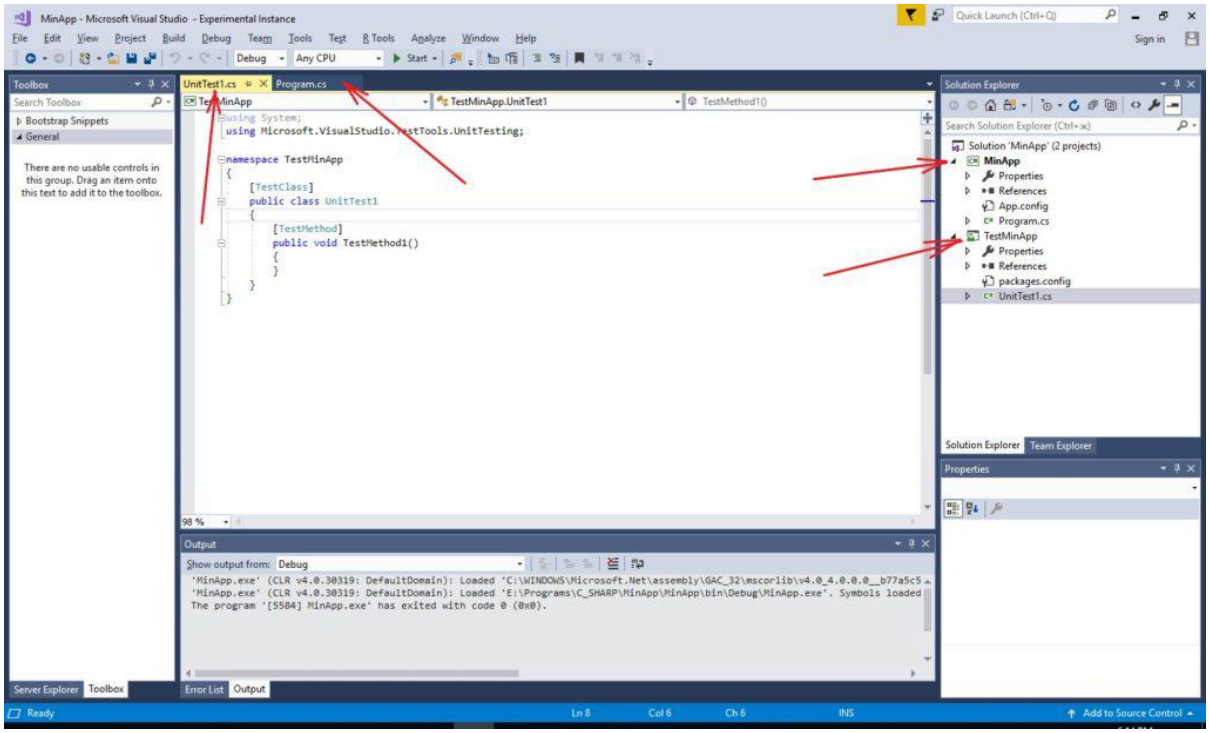

Рис. 4. Текст модуля UnitTest1.cs. Окно утилиты Solution Explorer с отображенными проектами TestMinApp и MinApp

4.2. Структура решения

Как видно из рисунка 4, утилита Solution Explorer отображает структуру решения (Solution Items), которое содержит два проекта:

- проект MinApp. Это проект, созданный по шаблону Console Application с функцией Min(), которую нужно протестовать;
- проект TestMinApp. Этот проект предназначен для тестирования функций проекта MinApp. Программный код, который тестирует функцию Min(), будет вноситься в файл проекта UnitTest1 проекта TestMinApp.

Оба проекта могут выполняться независимо друг от друга.

4.3. Текст файла «UnitTest1.cs». Атрибуты [TestMethod] и [TestClass]

В проекте TestMinApp главный интерес представляет файл теста UnitTest1.cs. В этом файле размещаются методы, которые будут тестировать функции проекта MinApp. Проект TestMinApp может содержать любое количество файлов, которые содержат тесты (например, UnitTest2.cs, UnitTest3.cs и т.д.).

Листинг файла UnitTest1.cs, сформированный MS Visual Studio, следующий:

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace TestMinApp
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMethod1()
        {
        }
    }
}
```

Как видно из вышеприведенного кода, файл содержит класс с именем UnitTest1. В классе есть общедоступный (public) метод с именем TestMethod1(). Перед реализацией метода TestMethod1() размещается атрибут [TestMethod]. Это означает, что в тело метода нужно вписать код, который будет тестировать функции проекта MinApp.

В классе можно вписывать любое количество методов, которые будут тестировать различные функции из разных модулей. Главное, чтобы эти методы были помечены атрибутом [TestMethod].

4.4. Выполнение изменений в тексте модуля UnitTest1. Изменение названия тестирующего метода

Допускается изменять названия методов и добавлять новые методы, которые помечены атрибутом [TestMethod] в модуле UnitTest1.cs. Учитывая это, в тексте модуля UnitTest1.cs нужно метод TestMethod1() переименовать на TestMin().

После выполненных изменений, сокращенный текст модуля файла UnitTest1.cs будет иметь вид:

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace TestMinApp
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMin()
        {
        }
    }
}
```

4.5. Подключение проекта MinApp к проекту TestMinApp

Чтобы иметь доступ к функции Min() (проект MinApp) из проекта TestMinApp, нужно подключить пространство имен в котором размещается эта функция.

Для этого прежде всего нужно вызвать контекстное меню для проекта TestMinApp. Затем в контекстном меню нужно вызвать команду «Add Reference…» (рисунок 5).

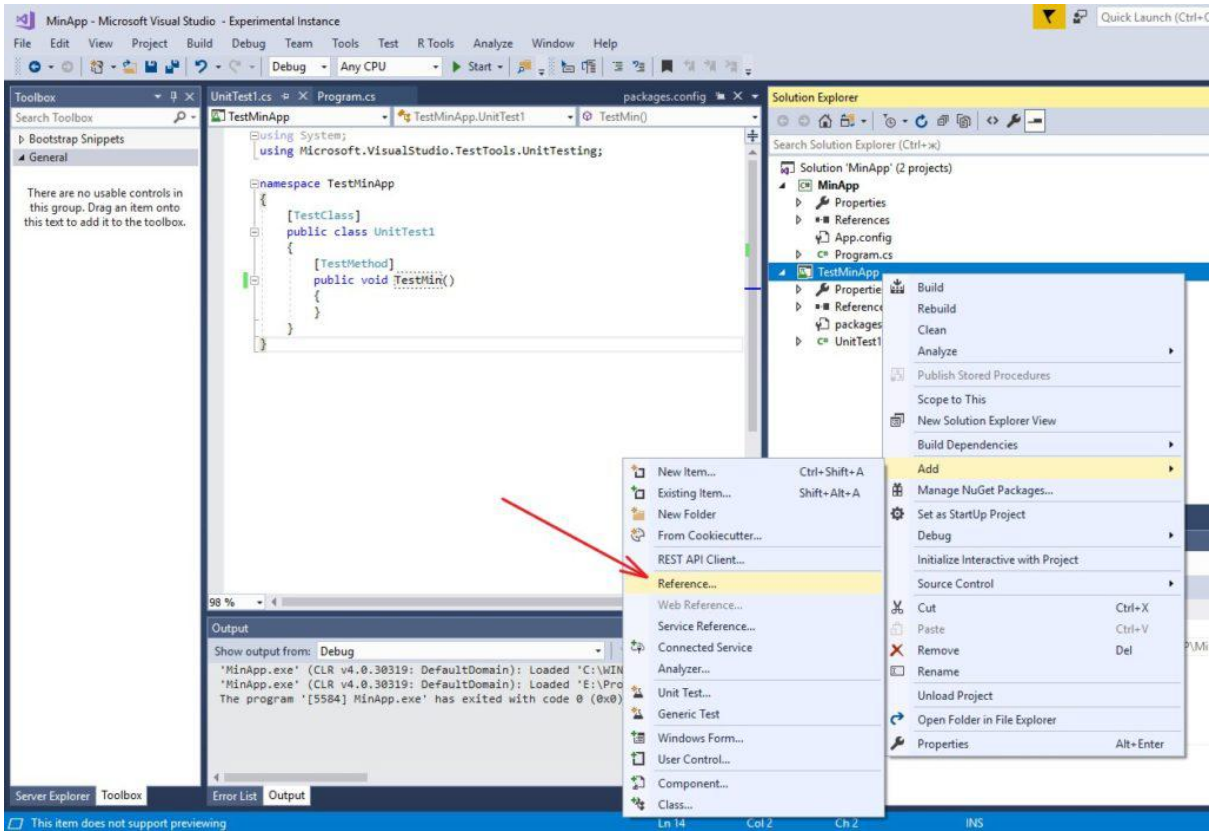

Рис. 5. Команда «Add Referencse…»

В результате откроется окно «Add Reference», в котором нужно:

- активировать вкладку Projects;
- в вкладке Projects выбрать проект MinApp.

Рис. 6. Окно «Add Reference». Подключение проекта MinApp

После выполненных действий функции проекта MinApp будут доступны для использования в проекте TestMinApp.

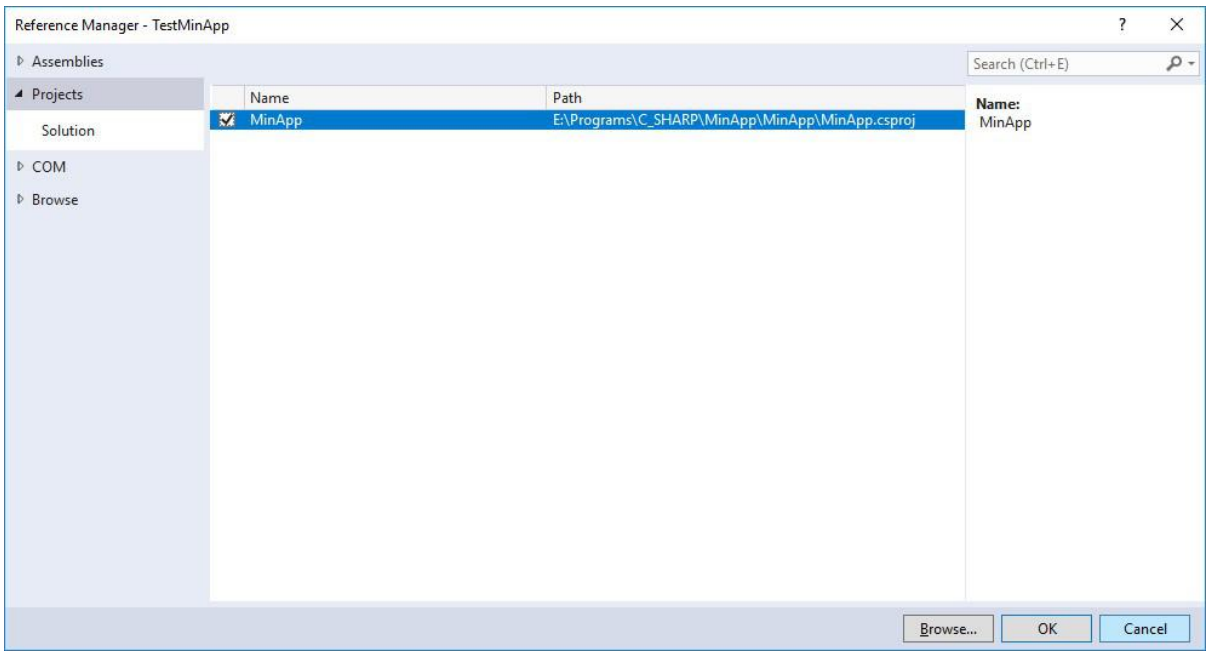

Рис. 6. Окно «Add Reference». Подключение проекта MinApp

После выполненных действий функции проекта MinApp будут доступны для использования в проекте TestMinApp.

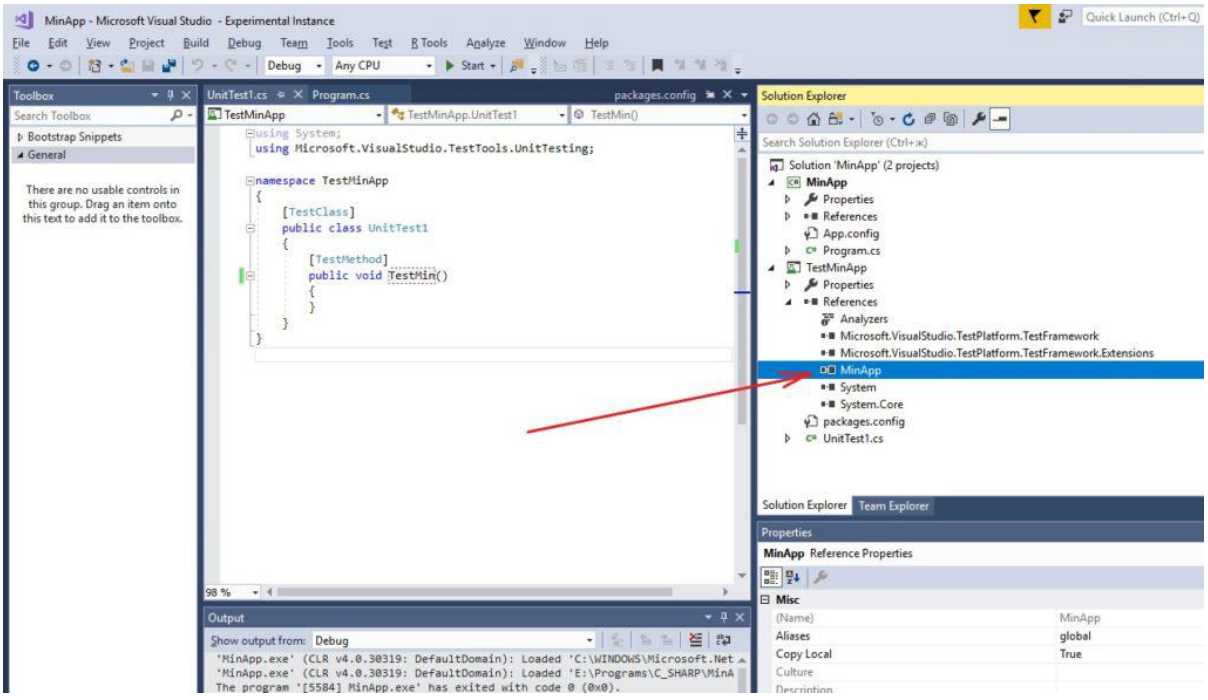

Рис. 7. Вкладка References с подключенным проектом MinApp

4.6. Внесение изменений в текст модуля UnitTest1.cs

4.6.1. Добавление пространства имен MinApp в модуль UnitTest1.cs

Добавить пространство имен MinApp с помощью директивы **using**:

```
using System;
using System.Text;
using System.Collections.Generic;
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;

using MinApp;

namespace TestMinApp
{
    ...
}
```

4.6.2. Текст метода TestMin()

В тексте метода TestMin() нужно ввести следующий код:

```
...

[TestClass]
public class UnitTest1
{
    [TestMethod]
    public void TestMin()
    {
        int min;
        min = Program.Min(3, 4, 5);
        Assert.AreEqual(2, min);
    }
}

...
```

4.7. Текст модуля UnitTest1.cs

Текст всего модуля UnitTest1.cs имеет следующий вид:

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;

using MinApp;

namespace TestMinApp
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void TestMin()
        {
            int min;
            min = Program.Min(3, 4, 5);
            Assert.AreEqual(2, min);
        }
    }
}
```

5. Запуск теста на выполнение и проверка результата тестирования

В Microsoft Visual Studio для работы с Unit-тестами реализовано специальное меню команд, которое называется Test.

Чтобы запустить тест на выполнение, нужно выбрать одну из команд

`Test -> Run -> Tests in Current Context`

или

`Test -> Run -> All Tests in Solution`

как показано на рисунке 8.

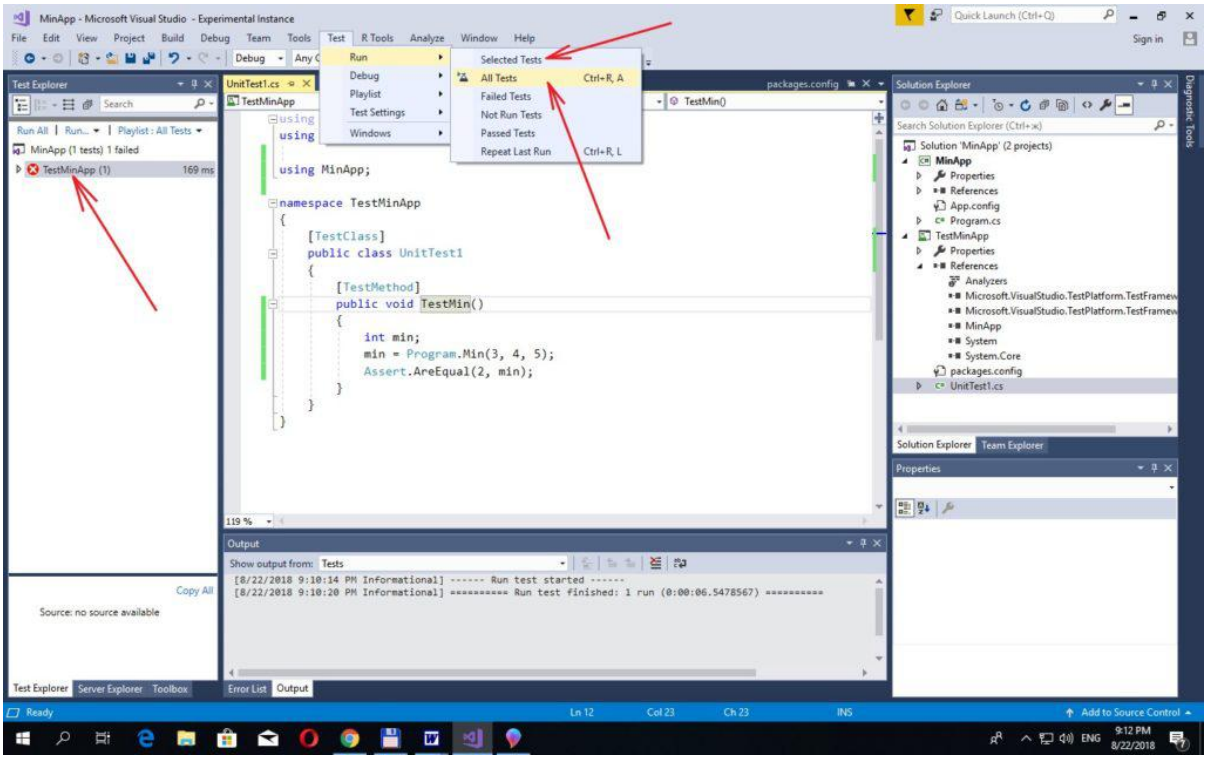

Рис. 8. Вызов команды запуска тестирования и просмотр результата

После запуска теста, результат можно просмотреть в левой части в окне Test Explorer. По всей видимости, тест не сдан. Это есть логично, так как в функции Assert.AreEqual() мы сравниваем числа 2 и 3, которые между собой различны. Здесь специально введено число 2 вместо числа 3.

Если вместо числа 2 ввести правильный ответ – число 3 (минимум между 3, 4, 5), то тест будет сдан. В этом случае текст метода TestMin() будет следующий:

```
...

[TestMethod]
public void TestMin()
{
    int min;
    min = Program.Min(3, 4, 5);
    Assert.AreEqual(3, min);
}

...
```

Соответственно в окне Test Explorer будет отображен положительный результат теста. После этого можно сделать вывод о том, что функция Min() для данного случая работает правильно.

6. Итог. Взаимодействие между проектами

В данной работе в решении (Solution) сформированы два проекта. Один проект MinApp содержит функцию Min(), которую нужно протестировать. Второй проект TestMinApp содержит методы, которые тестируют.

В Microsoft Visual Studio каждый из проектов запускается с помощью разных команд меню. Так, проект MinApp запускается стандартным способом из меню Run. А проект TestMinApp запускается из специального меню Test.