**[C++](https://www.bestprog.net/ru/category/cpp_ru/), [MS VISUAL STUDIO 2010](https://www.bestprog.net/ru/category/cpp_ru/%d0%bf%d1%80%d0%b0%d0%ba%d1%82%d0%b8%d0%ba%d0%b0-ru-2/ms-visual-studio-2010-ru/), [ПРАКТИКА](https://www.bestprog.net/ru/category/cpp_ru/%d0%bf%d1%80%d0%b0%d0%ba%d1%82%d0%b8%d0%ba%d0%b0-ru-2/)**

C++. ПРИМЕР СОЗДАНИЯ UNIT-ТЕСТА В MICROSOFT VISUAL STUDIO — C++.

[16 АВГУСТА, 2018](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/) [BESTPROGISCH](https://www.bestprog.net/ru/author/bestprogisch/)

**Пример создания Unit-теста в Microsoft Visual Studio — C++.**

В данной теме описывается пошаговый процесс создания простейшего Unit-теста в системе Microsoft Visual Studio 2010 (C++) для приложения типа CLR Console Application. Используя данный пример, можно научиться создавать собственные Unit-тесты. Пример также демонстрирует использование класса Assert для проведения тестирования работы функций.

**Содержание**

* [**Условие задачи**](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#task)
* [**Выполнение**](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#instr)
  + [1. Создать приложение по шаблону CLR Console Application](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p01)
  + [2. Реализация функции Max()](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p02)
  + [3. Текст программы, которую нужно протестировать](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p03)
  + [4. Создание теста](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p04)
    - [4.1. Добавление нового проекта к решению](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p04_1)
    - [4.2. Структура решения](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p04_2)
    - [4.3. Текст файла теста «UnitTest1.cpp». Атрибуты [TestMethod] и [TestClass]](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p04_3)
    - [4.4. Выполнение изменений в тексте модуля UnitTest1](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p04_4)
    - [4.5. Подключение модуля «MaxApp.cpp» проекта MaxApp к проекту TestMaxApp](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p04_5)
      * [4.5.1. Способ 1. Подключение с заданием полного имени](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p04_5_1)
      * [4.5.2. Способ 2. Подключение с заданием сокращенного имени](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p04_5_2)
    - [4.6. Написание программного кода для проведения тестирования в методе TestMax()](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p04_6)
  + [5. Запуск теста на выполнение и проверка результата тестирования](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p05)
  + [6. Текст модуля UnitTest1](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p06)
  + [7. Итог. Взаимодействие между проектами](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#p07)
* [**Связанные темы**](https://www.bestprog.net/ru/2018/08/16/example-of-unit-test-creating-in-microsoft-visual-studio-c_ru/#topics)

**Поиск на других ресурсах:**

Начало формы

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | |  |  | |  |

Конец формы

**Условие задачи**

Для приложения типа CLR Console Application необходимо разработать Unit-тест, который тестирует работу функции Max3(), определяющей максимальный элемент из трех чисел. Числа функция получает входными параметрами.

Для функции Max3() установить метод тестирования TestMax(). Проверить работу функции.

**Выполнение**

**1. Создать приложение по шаблону CLR Console Application**

После запуска Microsoft Visual Studio нужно выбрать

File->New Project...

В результате откроется окно New Project (рисунок 1), в котором нужно выбрать шаблон

Visual C++ -> CLR Console Application

Имя проекта задать MaxApp (или, по желанию, другое). В нашем случае папка для проекта установлена как «E:\Test\» (поле «Location:»).

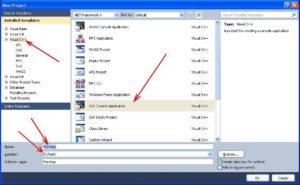
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_01.jpg)

Рис. 1. Создание приложения по шаблону CLR Console Application

После выбора OK, будет создано приложение типа CLR Console Application.

После создания приложения, окно текстовой части Microsoft Visual Studio будет иметь приблизительный вид, как показано на рисунке 2.

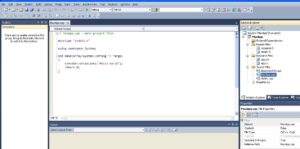
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_02.jpg)

Рис. 2. Вид приложения после создания

Как видно из рисунка 2, модуль MyApp.cpp содержит подключение модуля «stdafx.h» и функцию main(), которая есть точкой входа в программу (эта функция ассоциируется с программой на языке C++).

**2. Реализация функции Max()**

Перед объявлением   функции main() вводится текст функции Max(), который имеет следующий вид:

// функция, которая находит максимальное значение между тремя целыми числами

int Max(int a, int b, int c)

{

int max = a;

if (max<b) max = b;

if (max<c) max = c;

return max;

}

Именно эту функцию нужно будет протестировать с помощью Unit-теста в Microsoft Visual Studio.

**3. Текст программы, которую нужно протестировать**

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

// UnitTestApp01.cpp : Defines the entry point for the console application.

// MaxApp.cpp : main project file.

#include "stdafx.h"

using namespace System;

// функция, которая находит максимальное значение между тремя целыми числами

int Max(int a, int b, int c)

{

int max = a;

if (max<b) max = b;

if (max<c) max = c;

return max;

}

int main(array<System::String ^> ^args)

{

Console::WriteLine(L"Unit-test in MS Visual Studio.");

return 0;

}

Поскольку, эта программа будет тестироваться из другого модуля тестирования, то в функции main() большее ничего вводить не нужно. Так как, в соответствии с условием задачи, нужно протестовать работу функции Max(). 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 тестируемой программы. В нашем случае программа-тест будет вызывать функцию

int Max(int, int, int);

**4.1. Добавление нового проекта к решению**

Для создания теста нужно создать новый проект в решении (Solution). Для этого в Visual Studio вызывается последовательность команд

File -> Add -> New Project ...

В результате откроется окно «Add New Project», изображенное на рисунке 3.

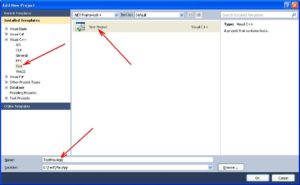
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_03.jpg)

Рис. 3. Окно «Add New Project»

В окне выбирается группа шаблонов Visual C++ — Test. Из отображаемых шаблонов выбирается шаблон проекта «Test Project«. В поле «Name» указывается имя проекта, который будет тестировать нашу программу. Нужно задать, например, TestMaxApp. Проект размещается в отдельной папке «E:\Test\MaxApp».

После выбора OK система создает новые файлы проекта которые будет тестировать, как показано на рисунке 4. Создается файл (модуль) с именем «UnitTest1.cpp». В этом файле нужно ввести программный код, который будет тестировать функцию Max() из модуля MaxApp.cpp.

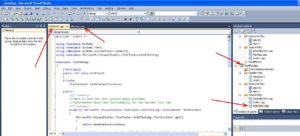
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_04.jpg)

Рис. 4. Текст модуля UnitTest1.cpp. Окно утилиты Solution Explorer с отображенными проектами TestMaxApp и MaxApp

**4.2. Структура решения**

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

* проект MaxApp. Это проект, созданный по шаблону CLR Console Application с функцией Max(), которую нужно протестовать;
* проект TestMaxApp. Этот проект предназначен для тестирования функций проекта MaxApp. Программный код, который будет тестировать функцию Max(), будет вноситься в файл проекта UnitTest1 проекта TestMaxApp.

Оба проекта могут выполняться независимо друг от друга.

**4.3. Текст файла теста «UnitTest1.cpp». Атрибуты [TestMethod] и [TestClass]**

В проекте TestMaxApp главным есть файл теста UnitTest1.cpp. В этом файле размещаются методы, которые будут тестировать функции проекта «MaxApp.cpp». Проект TestMaxApp может содержать любое количество файлов, которые содержат тесты (например, UnitTest2.cpp, UnitTest3.cpp и т.д.).

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

#include "stdafx.h"

using namespace System;

using namespace System::Text;

using namespace System::Collections::Generic;

using namespace Microsoft::VisualStudio::TestTools::UnitTesting;

namespace TestMaxApp

{

[TestClass]

public ref class UnitTest1

{

private:

TestContext^ testContextInstance;

public:

/// <summary>

///Gets or sets the test context which provides

///information about and functionality for the current test run.

///</summary>

property Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ TestContext

{

Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ get()

{

return testContextInstance;

}

System::Void set(Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ value)

{

testContextInstance = value;

}

};

#pragma region Additional test attributes

//

//You can use the following additional attributes as you write your tests:

//

//Use ClassInitialize to run code before running the first test in the class

//[ClassInitialize()]

//static void MyClassInitialize(TestContext^ testContext) {};

//

//Use ClassCleanup to run code after all tests in a class have run

//[ClassCleanup()]

//static void MyClassCleanup() {};

//

//Use TestInitialize to run code before running each test

//[TestInitialize()]

//void MyTestInitialize() {};

//

//Use TestCleanup to run code after each test has run

//[TestCleanup()]

//void MyTestCleanup() {};

//

#pragma endregion

[TestMethod]

void TestMethod1()

{

//

// TODO: Add test logic here

//

};

};

}

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 значит, что в тело метода нужно вписать код, который будет тестировать функции проекта MaxApp.

В классе можно вписывать любое количество методов, которые будут тестировать разные функции из разных модулей. Главное, чтобы эти методы были помечены атрибутом [TestMethod]. Например, если нужно добавить второй метод тестирования с именем MySecondTestMethod(), то в тело класса нужно вписать приблизительно следующий код:

[Test Method]

void MySecondTestMethod()

{

// тестирующий программный код метода MySecondTestMethod()

// ...

}

Аналогично, перед классом UnitTest1 размещается атрибут [TestClass]. Это означает, что в классе есть тестирующие методы.

**4.4. Выполнение изменений в тексте модуля UnitTest1**

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

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

#include "stdafx.h"

using namespace System;

using namespace System::Text;

using namespace System::Collections::Generic;

using namespace Microsoft::VisualStudio::TestTools::UnitTesting;

namespace TestProjectApp01

{

[TestClass]

public ref class UnitTest1

{

private:

TestContext^ testContextInstance;

public:

...

[TestMethod]

void TestMax()

{

//

// TODO: Add test logic here

//

};

};

}

**4.5. Подключение модуля «MaxApp.cpp» проекта MaxApp к проекту TestMaxApp**

Чтобы иметь доступ к функции Max() модуля «MaxApp.cpp» проекта MaxApp из проекта TestMaxApp, нужно вызвать этот модуль с помощью директивы #include.

Существует 2 способы такого подключения:

* подключить файл «MaxApp.cpp», указав его полное имя на диске (или на другом источнике);
* в ссылках на сборки Microsoft Visual Studio настроить папку с проектом MaxApp, чтобы она подключалась автоматически. После этого можно обращаться к файлу «MaxApp.cpp» по его сокращенному имени.

В данной теме описываются оба способа.

**4.5.1. Способ 1. Подключение с заданием полного имени**

Этот способ более упрощенный. Он есть удобным, когда нужно подключить в проект небольшое количество файлов для тестирования.

В проекте TestMaxApp в файле «UnitTest1.cpp» после подключения пространств имен нужно задать следующую строку:

#include "E:\\Test\\MaxApp\\MaxApp\\MaxApp.cpp"

Здесь указывается полное имя на диске файла MaxApp.cpp, в котором размещается функция Max(), которую нужно протестировать.

**4.5.2. Способ 2. Подключение с заданием сокращенного имени**

Этот способ эффективен, когда нужно протестовать несколько модулей разных проектов. Подключается целый каталог (папка) с файлами тестируемого проекта.

В нашем случае нужно подключить папку:

E:\Test\MaxApp\\MaxApp\

в перечень ссылок на сборки (проекты), которые автоматически подключаются к проекту TestMaxApp. После этого, можно будет подключать модуль MaxApp.cpp по сокращенному имени

#include "MaxApp.cpp"

избегая использования длинного имени как показано в пункте 4.5.1.

Прежде всего вызывается команда «References…» из контекстного меню TestMaxApp как показано на рисунке 5. Другой способ вызова – команда Project->References… (предварительно нужно выделить проект TestMaxApp).

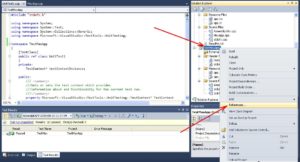
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_05.jpg)

Рис. 5. Вызов окна просмотра ссылок на сборки, которые используются в проекте TestMaxApp

В результате откроется окно «TestMaxApp Property Pages» которое изображено на рисунке 6.

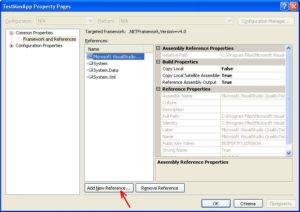
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_06.jpg)

Рис. 6. Окно TestMaxApp Property Pages, команда «Add New Reference…»

В окне «TestMaxApp Property Pages» сначала нужно активировать вкладку «Common Properties» -> «Framework and References».

Затем нужно выбрать команду «Add New Reference…». Эта команда разрешает добавлять в проект новые папки с модулями, методы (функции, ресурсы) которых потом можно включать в проект директивой #include. После выбора команды откроется окно Add Reference, изображенное на рисунке 7.

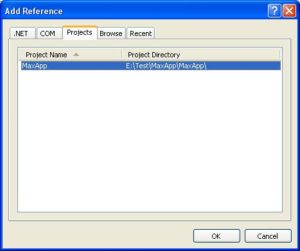
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_07.jpg)

Рис. 7. Окно Add Reference с отображенными проектами в текущем решении

В окне «Add Reference» во вкладке Project отображаются папки проектов, которые используются в данном решении (Solution). В нашем случае отображается только один проект MaxApp, который размещается в папке

E:\Test\MaxApp\MaxApp

После выбора на OK происходит возврат в предыдущее окно, в котором в перечне ссылок на сборки будет отображена ссылка на проект MaxApp.

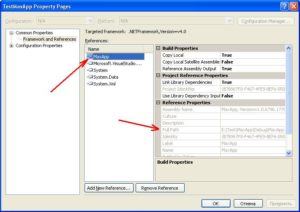
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_08.jpg)

Рис. 8. Окно «TestMaxApp Property Pages» после добавления проекта MaxApp в перечень общедоступных сборок

Это еще не все! Следующим шагом нужно подключить папку с проектом MaxTest в перечень «Include Directories».

Чтобы сделать это, нужно сначала активировать строку

Configuration Properties -> VC++ Directories

как показано на рисунке 9.

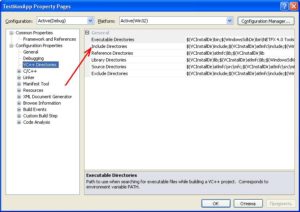
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_09.jpg)

Рис. 9. Окно TestMaxApp Property Pages, строка «Include Directories»

В перечне «General» выбирается «Include Directories». В результате раскрывается спадающий список, в котором нужно выбрать команду «<Edit…>». После этого откроется окно «Include Directories» изображенное на рисунке 10.

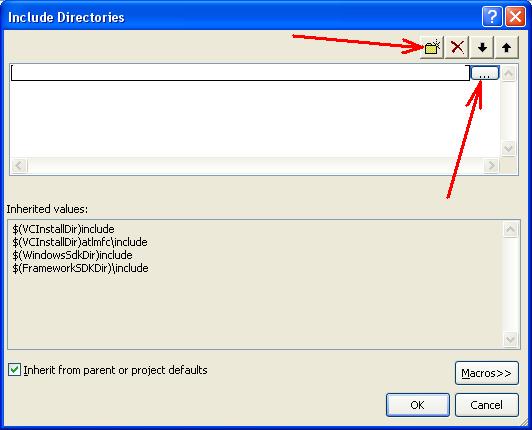
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_10.jpg)

Рис. 10. Команда «New Line» и кнопка выбора папки с файлами, которые подключаются

В окне «Include Directories» нужно добавить новую строку командой «New Line» и выбрать папку с проектом MaxApp, как показано на рисунке 10. После выбора кнопки «…» откроется стандартное окно Windows для выбора нужной папки. В нашем случае нужно выбрать папку

E:\Test\MaxApp\MaxApp

После выбора, окно Include Directories будет иметь вид как показано на рисунке 11.

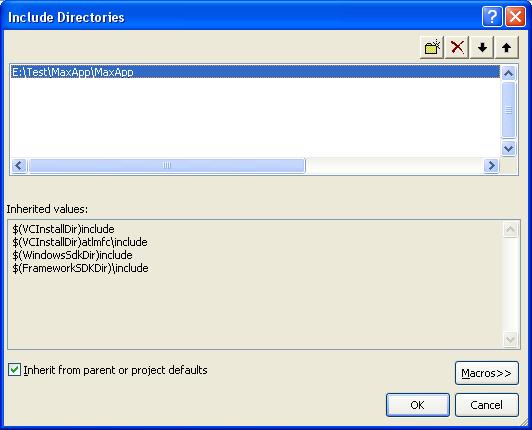
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_11.jpg)

Рис. 11. Окно Include Directories после выбора папки E:\Test\MaxApp\MaxApp

Для перехода к предыдущему окну нужно выбрать OK.

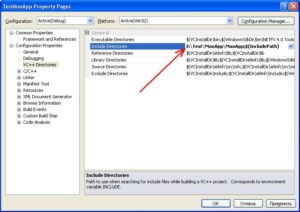
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_12.jpg)

Рис. 12. Окно TextMaxApp Property Pages после настройки строки «Include Directories»

Для перехода к написанию программного кода тестирования нужно выбрать OK.

После выполненных действий, все файлы из папки

E:\Test\MaxApp\MaxApp

можно подключать по сокращенному имени, например:

#include "MaxApp.cpp"

**4.6. Написание программного кода для проведения тестирования в методе TestMax()**

Программный код, который тестирует функцию Max() вписывается в тело метода TestMax() модуля «UnitTest1.cpp». Фрагмент программного кода метода TestMax() выглядит следующим образом:

...

[TestMethod]

void TestMax()

{

//

// TODO: Add test logic here

//

int t;

t = Max(5, 6, 7);

Microsoft::VisualStudio::TestTools::UnitTesting::Assert::AreEqual(t, 6);

};

...

В вышеприведенном коде вызывается функция AreEqual() из класса Assert. Эта функция сравнивает значение, которое было возвращено из функции Max() и число 6. В данном случае, происходит сравнение числа 7 (максимум) с числом 6. Для такого варианту тест не будет выполняться, так как 7 не равняется 6. В результате функция Assert::AreEqual() выбросит исключительную ситуацию (exception), что будет отображаться в специальном окне (см. п.5).

**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 изображено на рисунке 13.

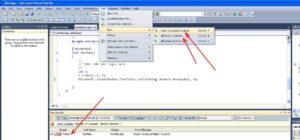
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_13.jpg)

Рис. 13. Вызов команды запуска тестирования и просмотр результата

После запуска теста, результат можно просмотреть в нижней части в окне Test Results. Как видно, тест не сдан. Это логично, поскольку в функции Assert::AreEqual() мы сравниваем числа 6 и 7, которые различны между собой. Здесь специально введено число 6 вместо числа 7.

Если вместо числа 6 ввести правильный ответ – число 7 (максимум между 5, 6, 7), то тест будет сдан. В этом случае текст метода TestMax() будет следующей:

...

[TestMethod]

void TestMax()

{

//

// TODO: Add test logic here

//

int t;

t = Max(5, 6, 7);

Microsoft::VisualStudio::TestTools::UnitTesting::Assert::AreEqual(t, 7);

};

...

Окно результата изображено на рисунке 14.

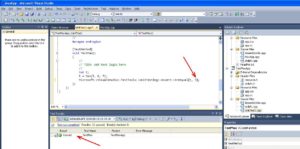
[](https://www.bestprog.net/wp-content/uploads/2018/08/05_02_00_011_14.jpg)

Рис. 14. Результат тестирования для случая, если ввести правильную проверку

Теперь можно сделать вывод о том, что функция Max() разработана правильно

**6. Текст модуля UnitTest1**

Ниже приводится текст модуля UnitTest1.cpp, тестирующий функцию Max() из модуля «MaxApp.cpp»:

#include "stdafx.h"

using namespace System;

using namespace System::Text;

using namespace System::Collections::Generic;

using namespace Microsoft::VisualStudio::TestTools::UnitTesting;

// способ 1 - задание полного имени

// #include "E:\\Test\\MaxApp\\MaxApp\\MaxApp.cpp"

// способ 2 подключение MaxApp по сокращенному имени

#include "MaxApp.cpp"

namespace TestMaxApp

{

[TestClass]

public ref class UnitTest1

{

private:

TestContext^ testContextInstance;

public:

/// <summary>

///Gets or sets the test context which provides

///information about and functionality for the current test run.

///</summary>

property Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ TestContext

{

Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ get()

{

return testContextInstance;

}

System::Void set(Microsoft::VisualStudio::TestTools::UnitTesting::TestContext^ value)

{

testContextInstance = value;

}

};

#pragma region Additional test attributes

//

//You can use the following additional attributes as you write your tests:

//

//Use ClassInitialize to run code before running the first test in the class

//[ClassInitialize()]

//static void MyClassInitialize(TestContext^ testContext) {};

//

//Use ClassCleanup to run code after all tests in a class have run

//[ClassCleanup()]

//static void MyClassCleanup() {};

//

//Use TestInitialize to run code before running each test

//[TestInitialize()]

//void MyTestInitialize() {};

//

//Use TestCleanup to run code after each test has run

//[TestCleanup()]

//void MyTestCleanup() {};

//

#pragma endregion

[TestMethod]

void TestMax()

{

//

// TODO: Add test logic here

//

int t;

t = Max(5, 6, 7);

Microsoft::VisualStudio::TestTools::UnitTesting::Assert::AreEqual(t, 7);

};

};

}

**7. Итог. Взаимодействие между проектами**

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

В Microsoft Visual Studio каждый из проектов запускается с помощью различных команд меню. Так, проект MaxApp запускается стандартным способом из меню Run. А тестирующий проект TestMaxApp запускается из специального меню Test.