

# **МЕТОДИЧЕСКИЕ РЕКОМЕНДАЦИИ**

по выполнению практических работ

**МДК 01.02 «Поддержка и тестирование программных модулей»**

по специальности

**09.02.07 Информационные системы и программирование**

очной формы обучения

## Содержание

Общая характеристика междисциплинарного курса.....	3
Тема 2.1. Отладка программных модулей .....	10
Практическая работа №1. Разработка и отладка модуля обработки элементов массива .....	10
Практическая работа №2. Разработка и отладка модуля обработки записей текстового файла .....	24
Тема 2.2. Отладка и тестирование программного продукта на уровне модулей .....	28
Практическая работа №3. Организация автоматизированного тестирования .....	28
Практическая работа №4. Создание модульных тестов .....	29
Практическая работа №5. Разработка системы тестов на основе потока управления .....	29
Лабораторная работа №1. Тестирование программного модуля по ранее определенному сценарию.....	30
Лабораторная работа №2. Тестирование с помощью инструментов среды разработки Visual Studio. Использование IntelliTest .....	31
Литература .....	34

## **Общая характеристика междисциплинарного курса**

В результате изучения профессионального модуля обучающийся должен освоить основной вид деятельности Разработка модулей программного обеспечения для компьютерных систем и соответствующих ему общих и профессиональных компетенций.

В результате освоения профессионального модуля студент должен:

иметь практический опыт в

- разработке кода программного продукта на основе готовой спецификации на уровне модуля;
- использовании инструментальных средств на этапе отладки программного продукта;
- проведении тестирования программного модуля по определенному сценарию;
- разработке мобильных приложений

уметь

- осуществлять разработку кода программного модуля на языках низкого и высокого уровней;
- создавать программу по разработанному алгоритму как отдельный модуль;
- выполнять отладку и тестирование программы на уровне модуля;
- осуществлять разработку кода программного модуля на современных языках программирования;
- уметь выполнять оптимизацию и рефакторинг программного кода;
- оформлять документацию на программные средства

знать

- основные этапы разработки программного обеспечения;
- основные принципы технологии структурного и объектно-ориентированного программирования;

- способы оптимизации и приемы рефакторинга;
- основные принципы отладки и тестирования программных продуктов

*Планируемые результаты освоения МДК:*

<b>Код</b>	<b>Наименование результата обучения</b>
ПК 1.1	Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием
ПК 1.2	Разрабатывать программные модули в соответствии с техническим заданием
ПК 1.3	Выполнять отладку программных модулей с использованием специализированных программных средств
ПК 1.4	Выполнять тестирование программных модулей
ПК 1.5	Осуществлять рефакторинг и оптимизацию программного кода
ПК 1.6	Разрабатывать модули программного обеспечения для мобильных платформ
ОК 01	Выбирать способы решения задач профессиональной деятельности, применительно к различным контекстам.
ОК 02	Осуществлять поиск, анализ и интерпретацию информации, необходимой для выполнения задач профессиональной деятельности.
ОК 03	Планировать и реализовывать собственное профессиональное и личностное развитие.
ОК 04	Работать в коллективе и команде, эффективно взаимодействовать с коллегами, руководством, клиентами.
ОК 05	Осуществлять устную и письменную коммуникацию на государственном языке с учетом особенностей социального и культурного контекста.

ОК 06	Проявлять гражданско-патриотическую позицию, демонстрировать осознанное поведение на основе общечеловеческих ценностей.
ОК 07	Содействовать сохранению окружающей среды, ресурсосбережению, эффективно действовать в чрезвычайных ситуациях.
ОК 08	Использовать средства физической культуры для сохранения и укрепления здоровья в процессе профессиональной деятельности и поддержания необходимого уровня физической подготовленности.
ОК 09	Использовать информационные технологии в профессиональной деятельности.
ОК 10	Пользоваться профессиональной документацией на государственном и иностранном языке.
ОК 11	Планировать предпринимательскую деятельность в профессиональной сфере.

## Тематический план и содержание раздела

<b>Раздел 2. Технологии тестирования программных модулей</b>		<b>102</b>
<b>МДК 01.02 Поддержка и тестирование программных модулей</b>		<b>102</b>
<b>Тема 2.1. Отладка программных модулей</b>	<i><b>Содержание</b></i>	
	1	Понятие отладки. Виды ошибок
	2	Инструменты отладки. Точка останова. Быстрые клавиши прерываний. Пошаговая отладка
	3	Отладочные классы
	4	Встроенные отладчики. Внешние отладчики
	5	Использование и документирование отладочной информации
	<b>Практические занятия</b>	
	1	Разработка и отладка модуля обработки элементов массива
	2	Разработка и отладка модуля обработки записей текстового файла
	3	Разработка и отладка модуля для генерации конечной последовательности случайных чисел и символов
	4	Разработка, отладка и оптимизация модуля выполнения операций реляционной алгебры над множествами
		<b>10</b>
		<b>6</b>

<b>Тема 2.2. Отладка и тестирование программного продукта на уровне модулей</b>	<b>Содержание</b>		<b>22</b>
	1	Спецификация программного модуля. Выявление несоответствие результата выполнения модуля его спецификации	
	2	Рефакторинг программного кода. Методы организации рефакторинга и оптимизации кода.	
	3	Основные положения теории отладки и тестирования. Термины и определения теории тестирования. Виды ошибок и способы их определения.	
	4	Виды тестирования. Порядок разработки тестов. Аксиомы тестирования. Методы тестирования.	
	5	Тестирование на основе потока управления. Цель модульного тестирования.	
	6	Тестирование на основе потока данных. Анализ результатов тестирования программы	
	7	Признаки проблемного кода и быстрые способы поиска некачественного кода	
	8	Автоматизация тестирования Возможности среды разработки для тестирования приложений	
<b>Лабораторные занятия</b>			<b>4</b>

	1	Тестирование программного модуля по ранее определенному сценарию	
	2	Отладка и тестирование программы на уровне модуля. Анализ результатов тестирования	
	3	Тестирование с помощью инструментов среды разработки	
	<b>Практические занятия</b>		<b>4</b>
	1	Разработка системы тестов на основе потока управления	
	2	Разработка системы тестов на основе потока данных	
<b>Тема 2.3. Документирование программных средств</b>	<b><i>Содержание</i></b>		<b>14</b>
	1	Цели документирования. Классификация и назначение документации на ПС. Документирование в процессе разработки ПС.	
	2	Средства разработки технической документации. Технологии разработки документов. Автоматизация разработки технической документации. Автоматизированные средства оформления документации	
	3	Документирование программного обеспечения в соответствии с Единой системой программной документации.	
	<b>Лабораторные занятия</b>		<b>4</b>



	1	Оформление документации на программные средства с использованием инструментальных средств	
	<b>Практические занятия</b>		<b>6</b>
	1	Ознакомление с системой стандартов ЕСПД	
	2	Разработка технического задания по предметной области	
	3	Разработка программы и методики испытаний в соответствии ТЗ	
	4	Разработка руководства пользователя и программиста	
<b>Самостоятельная работа при изучении раздела 2</b> 1. Работа с конспектами лекций, учебной и специальной литературой. 2. Подготовка к практическим занятиям с использованием методических рекомендаций преподавателя, оформление результатов практических занятий, отчётов и подготовка к их защите. 3. Выполнение индивидуальных заданий.			<b>14</b>

## Тема 2.1. Отладка программных модулей

### Практическая работа №1. Разработка и отладка модуля обработки элементов массива

#### Отладка кода C# с помощью Visual Studio.

*Отладка приложения* обычно означает запуск и выполнение приложения с подключенным отладчиком. При этом в отладчике доступно множество способов наблюдения за выполнением кода. Можно пошагово перемещаться по коду и просматривать значения, хранящиеся в переменных, задавать контрольные значения для переменных, чтобы отслеживать изменение значений, изучать путь выполнения кода, просматривать выполнение ветви кода и т. д.

Цель: изучить

- Запуск отладчика и попадание в точки останова.
- Использование команд для пошагового выполнения кода в отладчике.
- Проверка переменных в подсказках к данным и окнах отладчика.
- Просмотр стека вызовов

### Создание проекта

Создать проект консольного приложения .NET Framework. Запустить Visual Studio 2019.

Если окно запуска не открыто, выберите **Файл > Окно запуска**.

1. На начальном экране выберите **Создать проект**.
2. В поле поиска окна **Создание проекта** введите *консоль*. Затем выберите **C#** в списке языков и **Windows** в списке платформ.

Применив фильтры языка и платформы, выберите шаблон **Консольное приложение (.NET Framework)** и нажмите кнопку **Далее**.

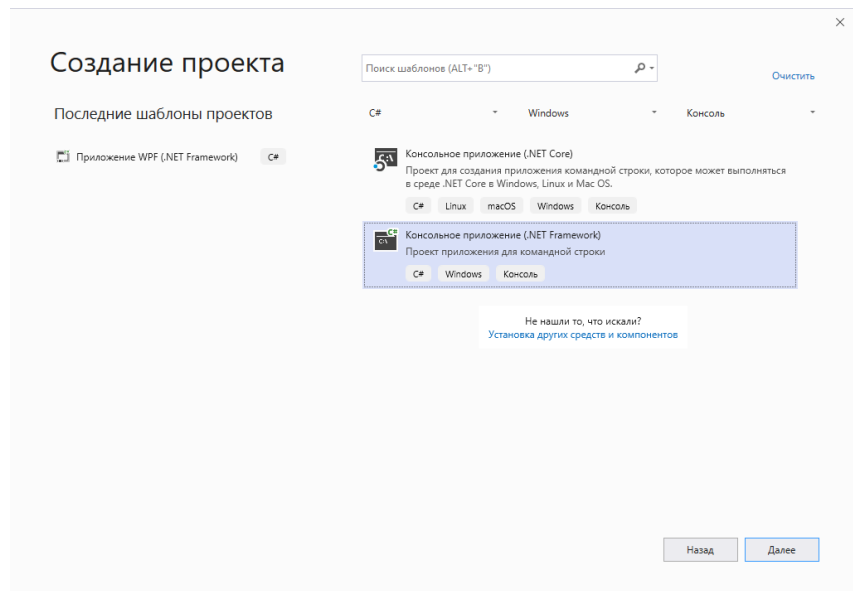


Рисунок 1

- В поле **Имя проекта** окна **Настроить новый проект** введите *GetStartedDebugging*, выберите расположение проекта. Затем нажмите **Создать**. Новый проект открывается в Visual Studio.

## Создание приложения

- Откройте файл *Program.cs* и замените все его содержимое по умолчанию следующим кодом:

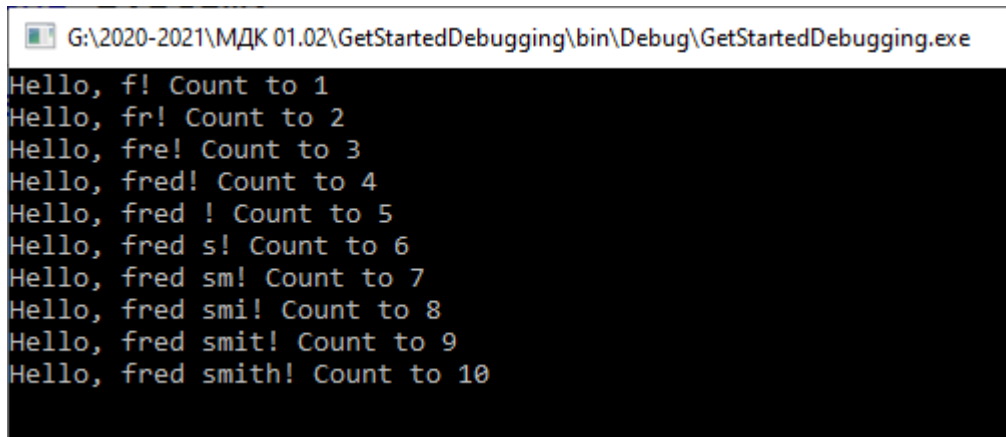
```
using System;
Ссылка: 0
class ArrayExample
{
    static void Main()
    {
        char[] letters = { 'f', 'r', 'e', 'd', ' ', 's', 'm', 'i', 't', 'h' };
        string name = "";
        int[] a = new int[10];
        for (int i = 0; i < letters.Length; i++)
        {
            name += letters[i];
            a[i] = i + 1;
            SendMessage(name, a[i]);
        }
        Console.ReadKey();
    }
    Ссылка: 1
    static void SendMessage(string name, int msg)
    {
        Console.WriteLine("Hello, " + name + "! Count to " + msg);
    }
}
```

Рисунок 2

## Запуск отладчика

1. Нажмите клавишу **F5** (**Отладка > Начать отладку**) или кнопку **Начать отладку** на панели инструментов отладки.

При нажатии клавиши **F5** происходит запуск приложения с присоединенным отладчиком. Но пока мы не сделали ничего особенного, чтобы проанализировать код. Поэтому приложение будет просто загружено, и вы увидите выходные данные консоли.



```
G:\2020-2021\МДК 01.02\GetStartedDebugging\bin\Debug\GetStartedDebugging.exe
Hello, f! Count to 1
Hello, fr! Count to 2
Hello, fre! Count to 3
Hello, fred! Count to 4
Hello, fred ! Count to 5
Hello, fred s! Count to 6
Hello, fred sm! Count to 7
Hello, fred smi! Count to 8
Hello, fred smit! Count to 9
Hello, fred smith! Count to 10
```

Рисунок 3

2. Остановите отладчик, нажав красную кнопку остановки ■ (**SHIFT + F5**).
3. В окне консоли нажмите клавишу, чтобы закрыть его.

## Установка точки останова и запуск отладчика

1. В цикле `for` функции `Main` установите точку останова, щелкнув левое поле следующей строки кода:

```
name += letters[i];
```

В месте установки точки останова появится красный круг ●.

Точки останова — это один из самых простых и важных компонентов надежной отладки. Точка останова указывает, где Visual Studio следует приостановить выполнение кода, чтобы вы могли проверить значения переменных или поведение памяти либо выполнение ветви кода.

2. Нажмите клавишу **F5** или кнопку **Начать отладку**. Запустится приложение и отладчик перейдет к строке кода, где задана точка останова.

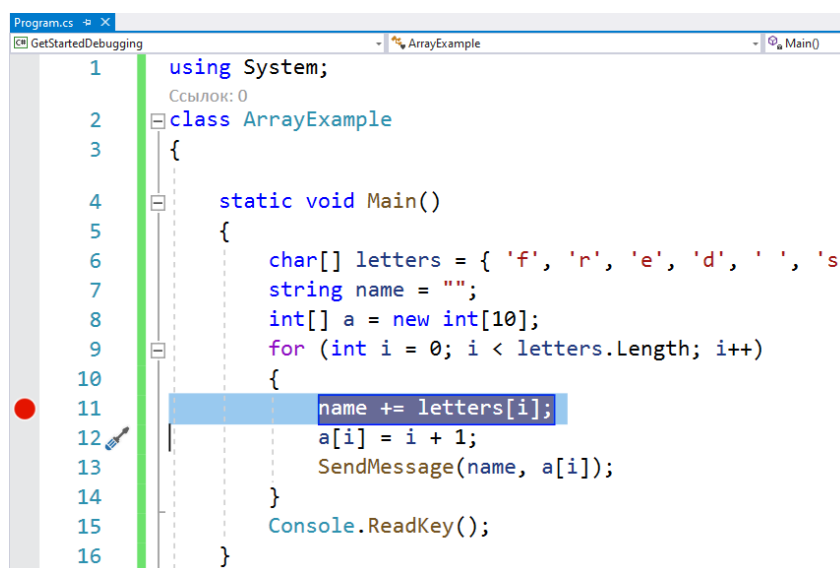


Рисунок 4

Желтая стрелка представляет оператор, на котором приостановлен отладчик. В этой же точке приостанавливается выполнение приложения (этот оператор пока не выполнен).

Если приложение еще не запущено, клавиша **F5** запускает отладчик и останавливается в первой точке останова. В противном случае **F5** продолжает выполнение приложения до следующей точки останова.

Точки останова полезны, если вам известны строка или раздел кода, которые вы хотите подробно изучить.

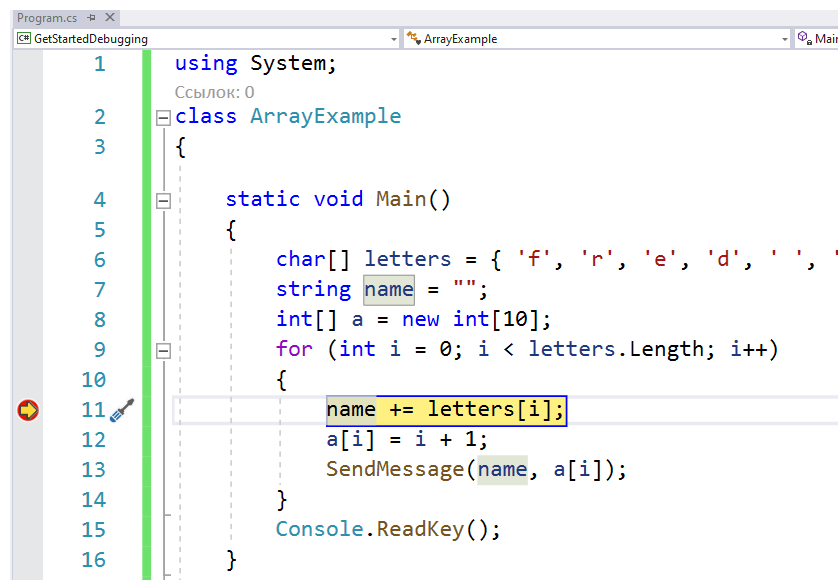


Рисунок 5

## Навигация по коду и проверка данных с помощью подсказок по данным

Здесь мы используем в основном сочетания клавиш, так как они позволяют быстро выполнять приложение в отладчике (эквивалентные команды, например команды меню, отображаются в круглых скобках).

1. При приостановке на операторе `name += letters[i]` наведите указатель мыши на переменную `letters` и увидите ее значение по умолчанию, а именно значение первого элемента в массиве — `char[0]`.

Функции, позволяющие проверять переменные, являются самыми полезными возможностями отладчика. Реализовывать эту задачу можно разными способами. Часто при попытке выполнить отладку проблемы пользователь старается выяснить, хранятся ли в переменных значения, которые требуются ему в определенное время.

2. Разверните переменную `letters`, чтобы просмотреть ее свойства, включая все элементы, которые она содержит.

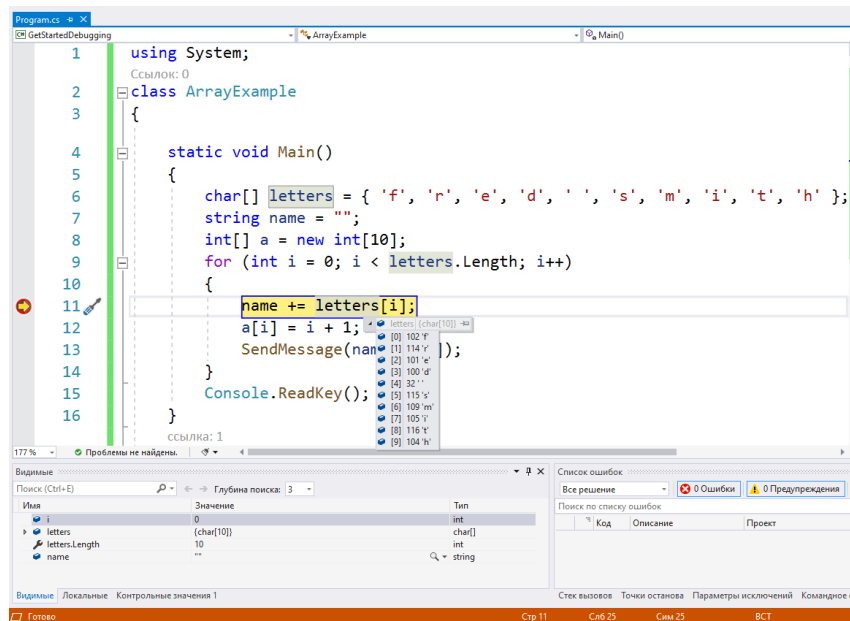


Рисунок 6

3. Затем наведите указатель мыши на переменную `name`, чтобы просмотреть ее текущее значение — пустую строку.
4. Нажмите клавишу **F10** (или выберите **Отладка > Шаг с обходом**) дважды, чтобы перейти к вызову метода `SendMessage`, а затем еще раз нажмите клавишу **F10**.

При нажатии клавиши **F10** отладчик переходит к следующей инструкции без захода в функции или методы в коде приложения (код продолжает выполняться). Нажимая клавишу **F10** в вызове метода `SendMessage`, мы пропускаем код реализации для `SendMessage` (который нас пока не интересует).

5. Несколько раз нажмите клавишу **F10** (или выберите **Отладка > Шаг с обходом**), чтобы выполнить несколько итераций по циклу `for`, каждый раз снова приостанавливая выполнение в точке останова и наводя указатель мыши на переменную `name`, чтобы просмотреть ее значение.

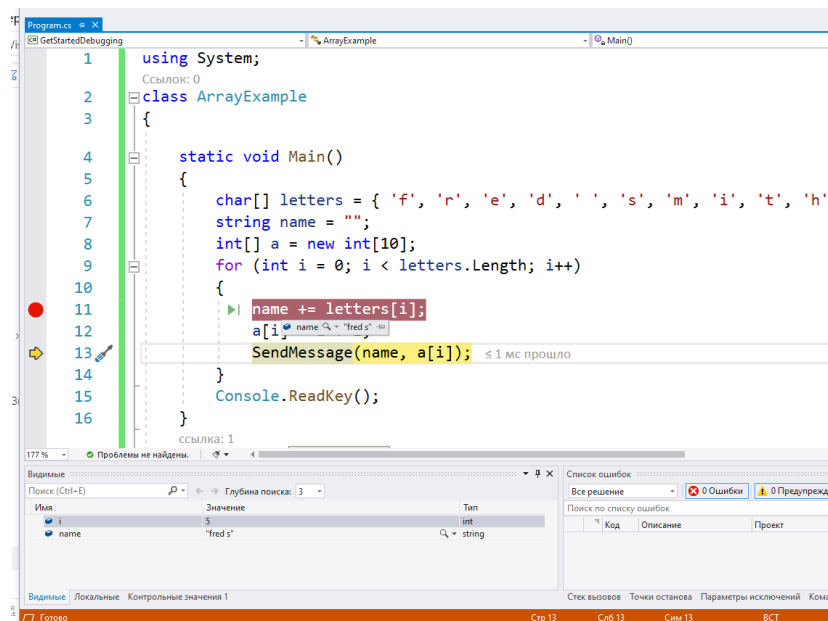


Рисунок 7

Значение переменной изменяется при каждой итерации цикла `for` — `f`, затем `fr`, `fre` и т. д. Для более быстрого прохода по циклу в этом сценарии можно нажать клавишу **F5** (или выбрать **Отладка > Продолжить**), чтобы перейти к точке останова, а не к следующей инструкции.

Часто при отладке требуется быстро проверить значения свойств в переменных, чтобы убедиться, что в них хранятся ожидаемые значения. Советы по данным — отличный способ это сделать.

6. Во время приостановки в цикле `for` в методе `Main` нажмите клавишу **F11** (или выберите **Отладка > Шаг с заходом**) несколько раз, пока не перейдете в вызов метода `SendMessage`.

Вы должны находиться в следующей строке кода:

```
SendMessage(name, a[i]);
```

7. Еще раз нажмите клавишу **F11**, чтобы выполнить шаг с заходом в метод `SendMessage`.



Желтый указатель перемещается в метод SendMessage.

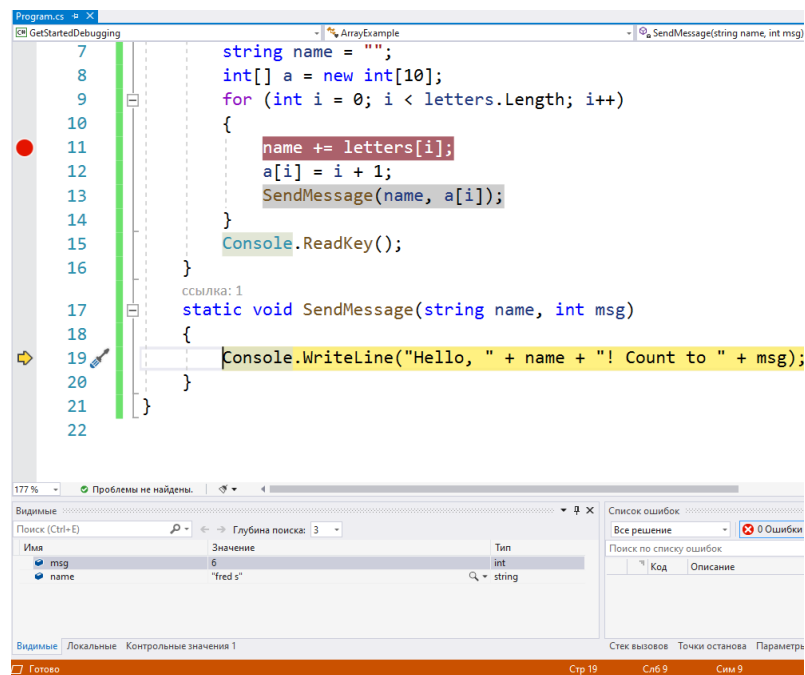


Рисунок 8

F11 — это команда **Шаг с заходом**, которая выполняет приложение с переходом к следующему оператору. Клавишу F11 удобно использовать для более детальной проверки потока выполнения. По умолчанию отладчик пропускает непользовательский код (дополнительные сведения см. в статье об [отладке в режиме "Только мой код"](#)).


Предположим, что вы закончили изучать метод SendMessage и хотите выйти из него, но остаться в отладчике. Это можно сделать с помощью команды **Шаг с выходом**.

8. Нажмите сочетание клавиш **SHIFT + F11** (или **Отладка > Шаг с выходом**).

Эта команда возобновляет выполнение приложения (и работу отладчика) до возврата данных текущим методом или текущей функции.

Вы должны вернуться в цикл for в методе Main, приостановленный на вызове метода SendMessage.

## Переход по коду с помощью команды "Выполнение до щелкнутого"

1. Нажмите клавишу **F5**, чтобы снова перейти к точке останова.
2. В редакторе кода прокрутите вниз и наведите указатель мыши на метод `Console.WriteLine` в методе `SendMessage`, чтобы в левой части появилась зеленая кнопка **Выполнение до щелкнутого** . В подсказке для кнопки выводится "Выполнить до этого места".

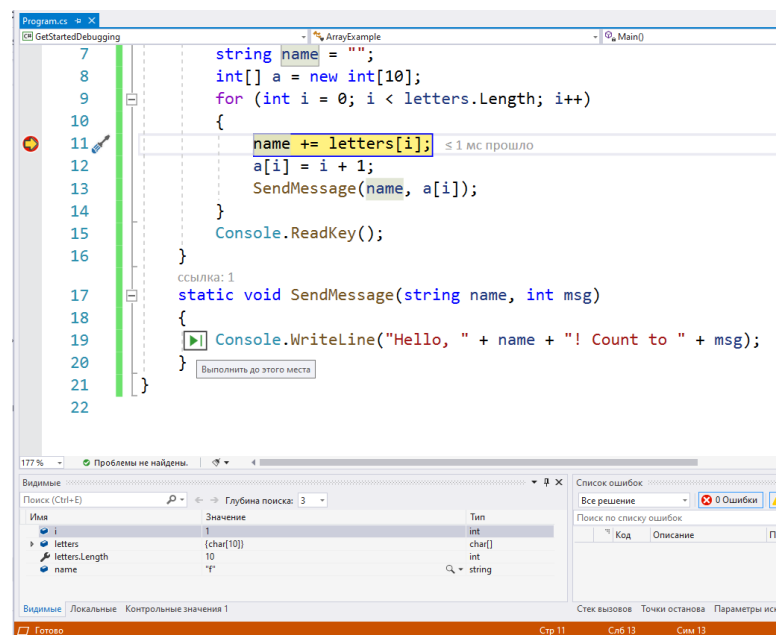



Рисунок 9

3. Нажмите кнопку **Выполнение до щелкнутого** .

Отладчик перемещается к методу `Console.WriteLine`.

Использование этой кнопки аналогично установке временной точки останова. Функция **Выполнение до щелкнутого** удобна для быстрой работы в видимой области кода приложения (можно щелкнуть в любом открытом файле).

## Быстрый перезапуск приложения

Нажмите кнопку **Перезапустить**  на панели инструментов отладки (**CTRL + SHIFT + F5**).

Кнопка **Перезапустить** позволяет сэкономить время, затрачиваемое на остановку приложения и перезапуск отладчика. Отладчик приостанавливается в первой точке останова, достигнутой при выполнении кода.

Отладчик еще раз останавливается в точке останова, ранее заданной вами в цикле `for`.

## Проверка переменных с помощью окон "Видимые" и "Локальные"

1. Взгляните на окно **Видимые** в нижней части редактора кода.

Если оно закрыто, откройте его во время приостановки в отладчике, выбрав **Отладка > Окна > Видимые**.

В окне **Видимые** отображаются переменные и их текущие значения. В окне **Видимые** отображаются все переменные, используемые в текущей или предыдущей строке (сведения о зависящем от языка поведении см. в соответствующей документации).

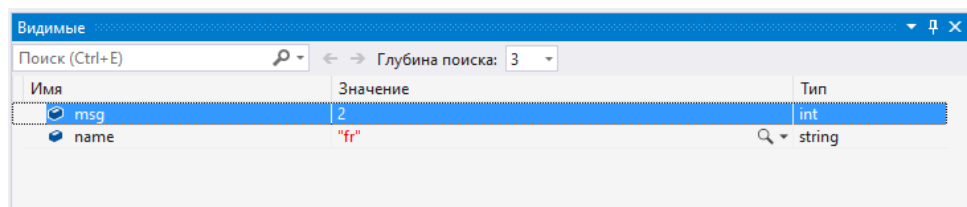
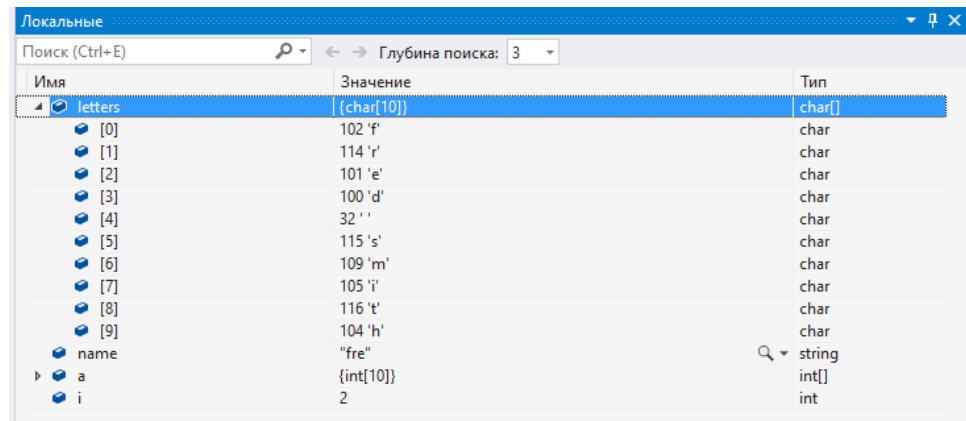


Рисунок 10

2. Затем посмотрите на окно **Локальные** на вкладке рядом с окном **Видимые**.

3. Разверните переменную `letters`, чтобы отобразить элементы, которые она содержит.



Имя	Значение	Тип
letters	{char[10]}	char[]
[0]	102 'f'	char
[1]	114 'r'	char
[2]	101 'e'	char
[3]	100 'd'	char
[4]	32 ' '	char
[5]	115 's'	char
[6]	109 'm'	char
[7]	105 'i'	char
[8]	116 't'	char
[9]	104 'h'	char
name	"fre"	string
a	{int[10]}	int[]
i	2	int

Рисунок 11

### Установка контрольного значения

1. В основном окне редактора кода щелкните правой кнопкой мыши переменную `name` и выберите команду **Добавить контрольное значение**.

В нижней части редактора кода откроется окно **Контрольное значение**. В окне **Контрольное значение** можно указать переменную (или выражение), которую необходимо отслеживать.

Теперь у вас есть контрольное значение, заданное для переменной `name`, и по мере перемещения по отладчику вы можете наблюдать за изменением его значения. В отличие от других окон переменных, в окне **Контрольное значение** всегда отображаются просматриваемые вами переменные (они выделяются серым цветом, когда находятся вне области действия).

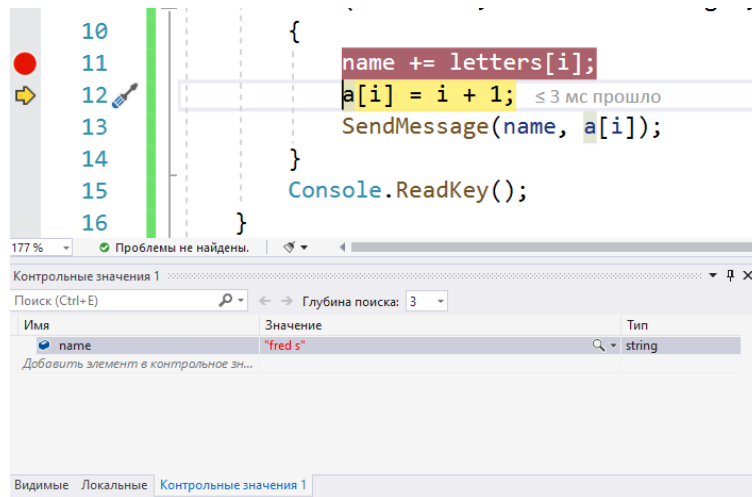


Рисунок 12

## Просмотр стека вызовов

1. Во время приостановки в цикле `for` щелкните окно **Стек вызовов**, которое по умолчанию открыто в нижней правой области. Если оно закрыто, откройте его во время приостановки в отладчике, выбрав **Отладка > Окна > Стек вызовов**.
2. Несколько раз нажмите клавишу **F11**, пока отладчик не приостановится в методе `SendMessage`. Взгляните на окно **Стек вызовов**.

В окне **Стек вызовов** показан порядок вызова методов и функций. В верхней строке приведена текущая функция (в данном приложении метод `SendMessage`). Во второй строке показано, что функция `SendMessage` была вызвана из метода `Main` и т. д.

Стек вызовов хорошо подходит для изучения и анализа потока выполнения приложения.

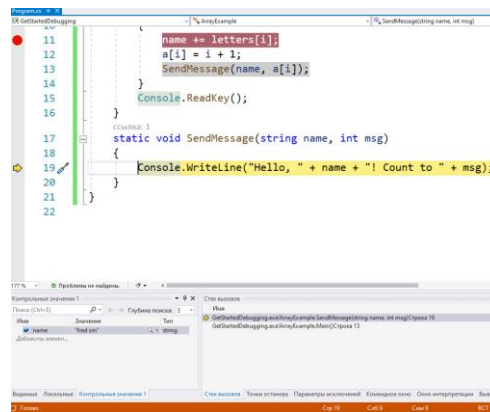


Рисунок 13

Для выполнения других задач можно воспользоваться контекстными меню из окна **Стек вызовов**. Например, можно вставлять точки останова в указанные функции, перемещать отладчик с помощью функции **Выполнение до текущей позиции** и изучать исходный код.

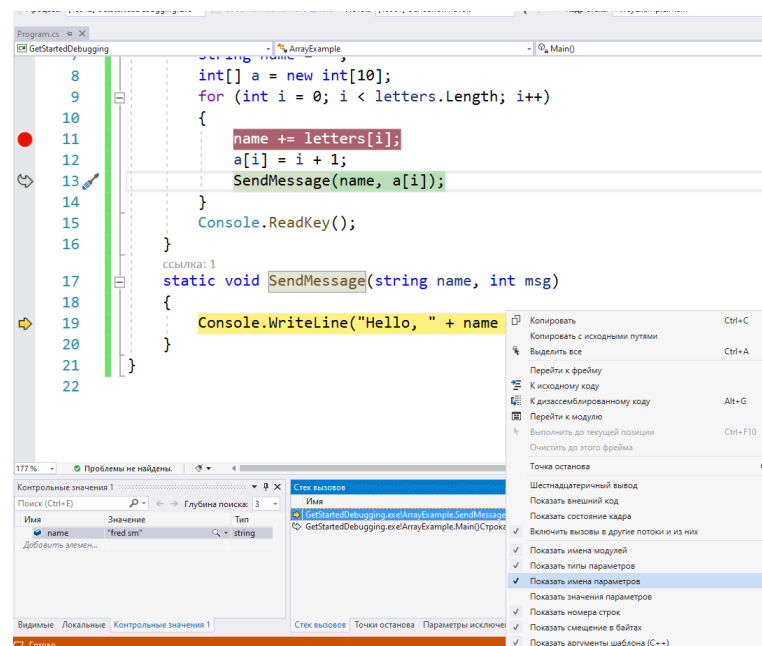


Рисунок 14

## Изменение потока выполнения

1. Дважды нажмите клавишу **F11**, чтобы запустить метод `Console.WriteLine`.
2. Приостановив отладчик в вызове метода `SendMessage`, с помощью мыши захватите желтую стрелку (указатель выполнения) в левой части и

переместите ее вверх на одну строку — обратно в `Console.WriteLine`.

### 3. Нажмите клавишу **F11**.

Отладчик повторно выполнит метод `Console.WriteLine` (вы увидите это в выходных данных окна консоли).

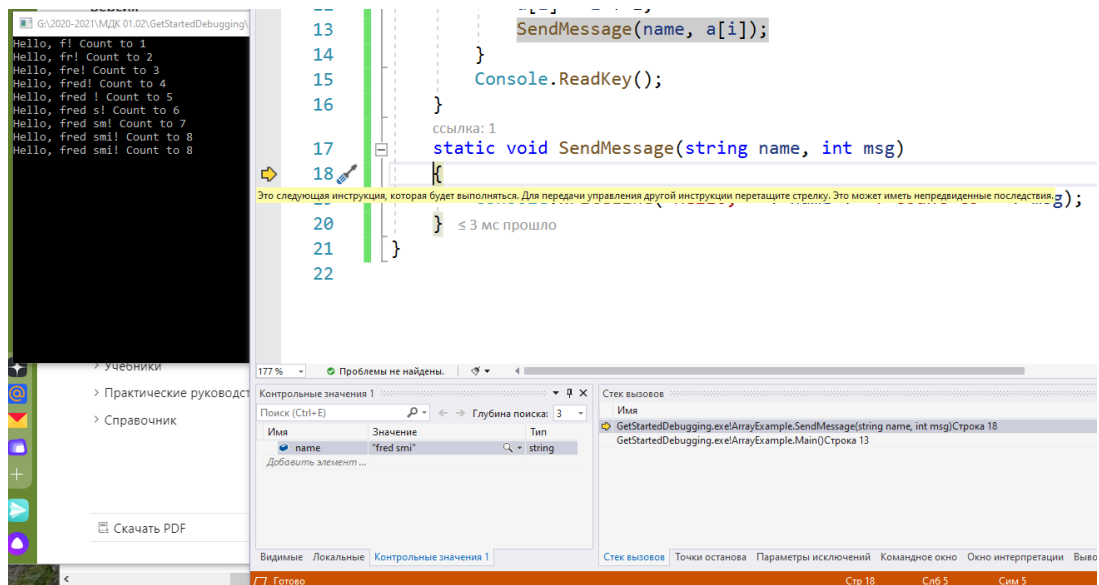


Рисунок 15

Изменяя поток выполнения, можно решать множество задач, например тестировать различные пути выполнения кода или повторно выполнять код без перезапуска отладчика.

## Предупреждение

Как правило, при работе с этой функцией необходимо соблюдать осторожность — вы увидите соответствующее предупреждение во всплывающей подсказке. Могут отображаться и другие предупреждения. При перемещении указателя предыдущее состояние приложения не возвращается.

### 4. Чтобы продолжить выполнение приложения, нажмите клавишу **F5**.

## *Задание*

*Условия задач по вариантам из практической работы №11  
Программирование с использованием одномерных массивов, МДК 01.01*

*Сдать на проверку архив с приложением, отчет в формате .docx*

### **Практическая работа №2. Разработка и отладка модуля обработки записей текстового файла**

Дефекты(Deffects). Основополагающие артефакты процесса тестирования – описывают обнаруженные факты несоответствия системы предъявляемым требованиям. Являются одним из подтипов запросов на изменение, описывающих найденную ошибку или несоответствие на всех этапах тестирования. Хотя базу данных дефектов можно вести в текстовом файле или Excel таблице, предпочтительным является использования специализированного инструментального средства, которое позволяет передавать информацию об обнаруженных дефектах от тестировщиков к разработчикам, а в обратную сторону – сведения об устранении дефектов. А также формировать необходимые отчеты о тенденциях изменения количества обнаруживаемых и устраняемых дефектов.

#### Модель работы с дефектами

Каждый обнаруженный дефект программного комплекса должен быть зарегистрирован и с ним должна быть проведена работа, которая в конечном итоге должна привести к исправлению дефекта. Работа с дефектами осуществляется по выбранной модели работы с дефектами. Выбор модели работы с дефектами зависит от конкретного проекта, структуры подразделения разработчиков и инженеров по тестированию. Для разработанной системы поддержки процесса тестирования была



спроектирована модель работы с дефектами, которая представлена на рисунке 5:

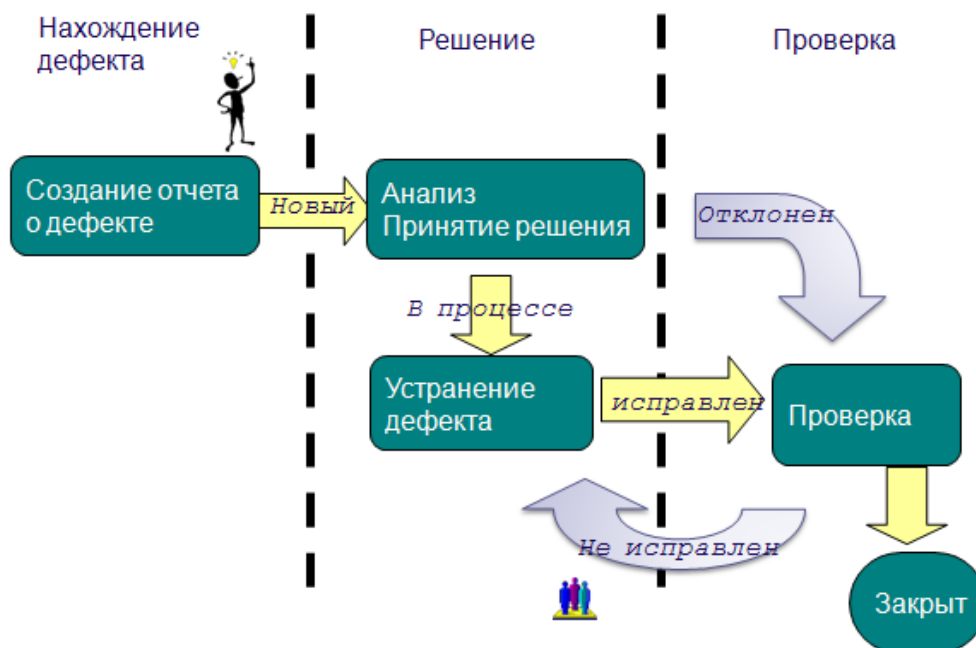


Рисунок 16

Как только дефект обнаружен, тестировщик регистрирует его в подсистеме работы с дефектами, присваивая ему статус «Новый» и направляет его на руководителя группы тестирования. Руководитель группы тестирования подтверждает дефект и направляет его на руководителя группы разработчиков или направляет дефект на инженера по тестированию на доработку. Далее руководитель группы разработчиков направляет дефект на студента или специалиста, отвечающего за дефект, студент или специалист изучает суть дефекта и проставляет статус «В процессе». Если в данный момент необходимо отложить исправление дефекта, то разработчик может присвоить дефекту статус «Отложен». После исправления дефекта разработчик присваивает ему статус «Исправлен» и направляет дефект на руководителя группы тестировщиков, а тот направляет дефект на тестировщика. Инженер по тестированию проверяет дефект и если он исправлен, присваивает ему статус «Исправлен». Если дефект не исправлен, то ему присваивается статус «Не исправлен» и направляется на студента или специалиста для исправления.

Всем дефектам со статусом «Исправлен» руководитель группы тестирования присваивает статус «Закрит».

Кроме статусов атрибутами дефекта являются *приоритет и важность*. Приоритет дефекта показывает насколько быстро нужно исправить дефект, по приоритету определяется очередность выполнения задачи. Важность характеризует критичность дефекта по отношению к тестируемому приложению. Например, по признаку «важность» дефект может быть описан как «блокирующий» - не позволяющий приложению запускаться, «критичный» - некоторые функции приложения недоступны, «крупный» - какая-то функция не работает, «средний» - часть функции не работает и «второстепенный» - не влияющий на работу приложения (например, неправильное написание или неудобное расположение элементов управления). Приоритет в свою очередь может быть «высоким», «средним» и «низким».

Такая модель работы с дефектами позволяет полностью контролировать состояние тестируемого проекта, распределяет роли между участниками процесса тестирования и делает процесс работы с дефектами открытым и отслеживаемым как для тестировщиков, так и для разработчиков.

Задание.

Разработать приложение в среде Visual Studio, технология WPF.

*Результаты сохранить в текстовый файл!*

1. (4 балла) Напишите программу с графическим пользовательским интерфейсом, в которой в два текстовых поля вводятся целые числа. После нажатия кнопки «Рассчитать» программа должна вычислить сумму, разность, частное и произведение введенных чисел и вывести результат каждой операции в отдельные поля. В случае попытки деления на 0 программа должна выводить какое-либо сообщение.

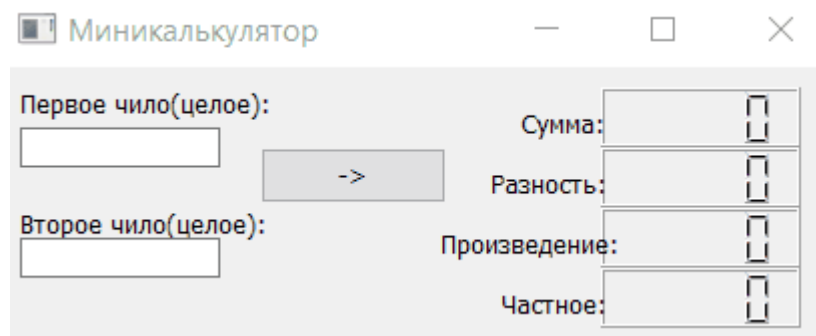


Рисунок 17

2. (4 балла) Напишите программу «Заказ в Макдональдсе» с графическим пользовательским интерфейсом. Пользователь должен иметь возможность выбирать одно или несколько блюд. После нажатия на кнопку «Заказать» в отдельном текстовом элементе (нередатируемом) должен отображаться «чек» с выбранными позициями.

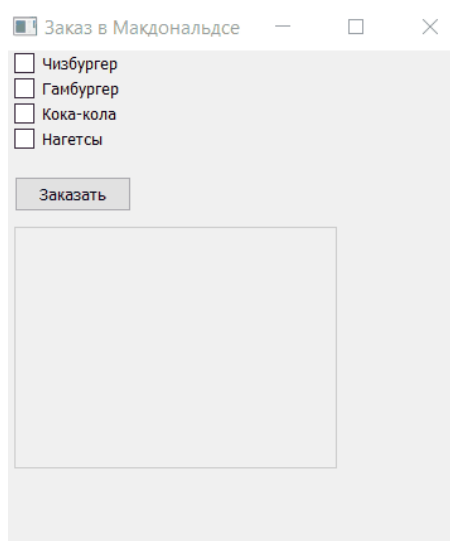


Рисунок 18

3. (5 баллов) Модифицируйте задачу «Заказ в Макдональдсе». Добавьте возможность указать не только блюдо, но и количество порций. У каждого блюда есть цена. По умолчанию, если блюдо выбрано, количество становится равным 1. В чеке должна быть отображена следующая информация: блюдо, количество, итоговая стоимость блюда каждого типа и суммарная стоимость заказа.

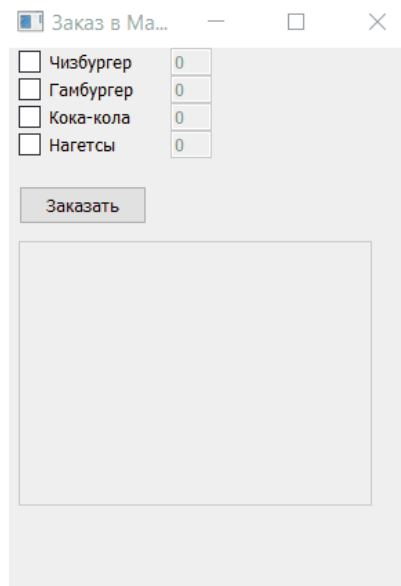


Рисунок 19

Заполнить таблицу:

№	Описание дефекта	Скрин	Статус	Приоритет	Важность

## Тема 2.2. Отладка и тестирование программного продукта на уровне модулей

### Практическая работа №3. Организация автоматизированного тестирования

Скачайте и установите программу TestComplete. Выберите приложение для тестирования (.exe файл).

Создайте модули с тестовыми функциями по описанию (пример по ссылке <https://habr.com/ru/sandbox/73760/>).

В ответ загрузите архив с полученным проектом и файл .exe, который тестировался.

#### **Практическая работа №4. Создание модульных тестов**

##### **Задание 1.**

Разработать модульные тесты для библиотеки MyCalc- по несколько для каждого метода.

Добавить в библиотеку MyCalc метод "Квадратный корень из числа".

Добавить в тестовый модуль соответствующую тестовую функцию. (см. пример Тест-метод Деление на ноль)

##### **Задание 2.**

Создайте проект для проверки пароля при регистрации пользователя.

Именованная всех компонентов сформируйте самостоятельно.

Реализуйте юнит-тест в этом же решении отдельным проектом, самостоятельно определив именованная.

Проверьте работоспособность проектов.

Сдать архивы с полученными проектами

#### **Практическая работа №5. Разработка системы тестов на основе потока управления**

Ход работы:

1. Создать консольное приложение для задачи по своему варианту
2. Построить управляющий граф программы.
3. Составить набор тестов в соответствии с критериями C0, C1, C2

4. Провести тестирование программы по тестовому набору
5. Оформить отчет: программный код, скрины выполнения, результаты тестирования в виде таблицы
6. Сдать отчет

14. Определить, сколько раз встречается минимальный элемент в последовательности.

15. Определить, сколько раз встречается максимальный элемент в последовательности.

16. Выбрать максимальный из модулей элементов последовательности.

17. Определить, сколько нулей в последовательности.

18. Осуществляя ввод элементов последовательности до тех пор, пока не будет введено заданное число, подсчитать их количество.

19. Напечатать True, если элементы последовательности упорядочены по возрастанию, и False — в противном случае.

20. В последовательности натуральных чисел подсчитать количество чисел, оканчивающихся заданной цифрой.

21. В заданной последовательности определить максимальное количество подряд идущих положительных чисел.

22. Найти сумму тех членов последовательности, которые оканчиваются на заданную цифру.

23. Найти сумму четных элементов последовательности целых чисел.

### **Лабораторная работа №1. Тестирование программного модуля по ранее определенному сценарию**

По заданию практической работы №5:

1. Разработать библиотеку класса по консольному приложению
2. Разработать соответствующий тестовый класс с тестовыми методами
3. Провести Unit-тестирование
4. Оформить отчет о выполненной работе
5. Сдать отчет и архив с проектом

## Лабораторная работа №2. Тестирование с помощью инструментов среды разработки Visual Studio. Использование IntelliTest

### Задание

1. Установить Visual Studio 2019 Enterprise Edition (бесплатная пробная версия 90 дней)
2. Создать проект библиотеки класса .NET Framework
3. Запустить IntelliTest для класса (для всех методов)
4. Изучить и проанализировать результаты тестирования
5. Оформить и сдать отчет о выполненной работе

### Работа с IntelliTest

1. Установить Visual Studio 2019 Enterprise Edition (бесплатная пробная версия 90 дней)

<https://visualstudio.microsoft.com/ru/downloads/>

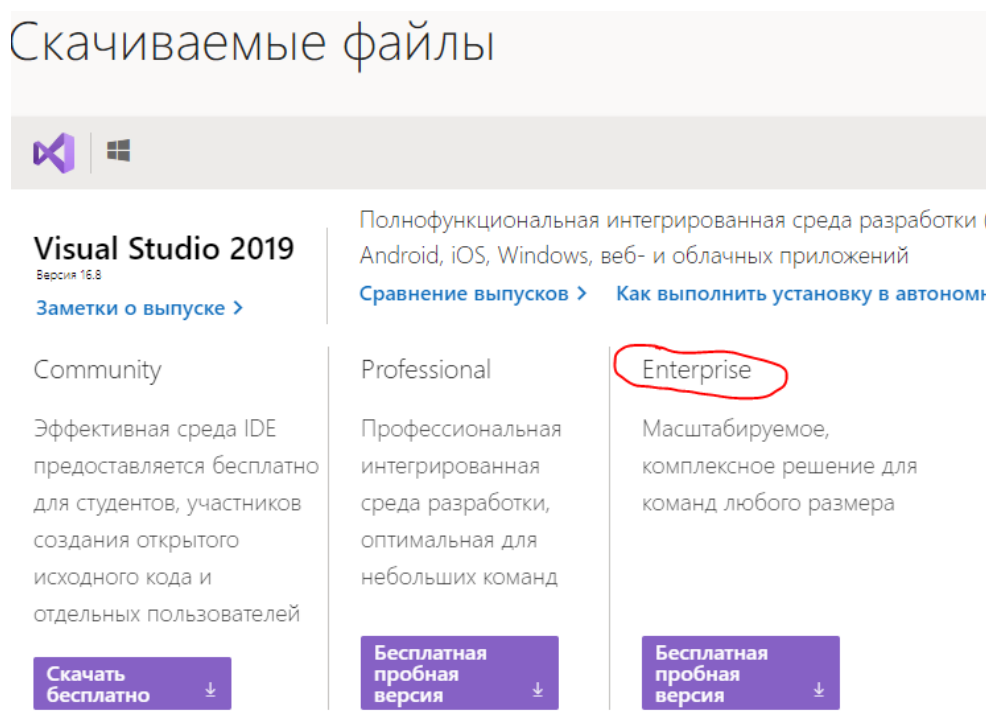


Рисунок 20

## 2. Создать проект библиотеки класса .NET Framework

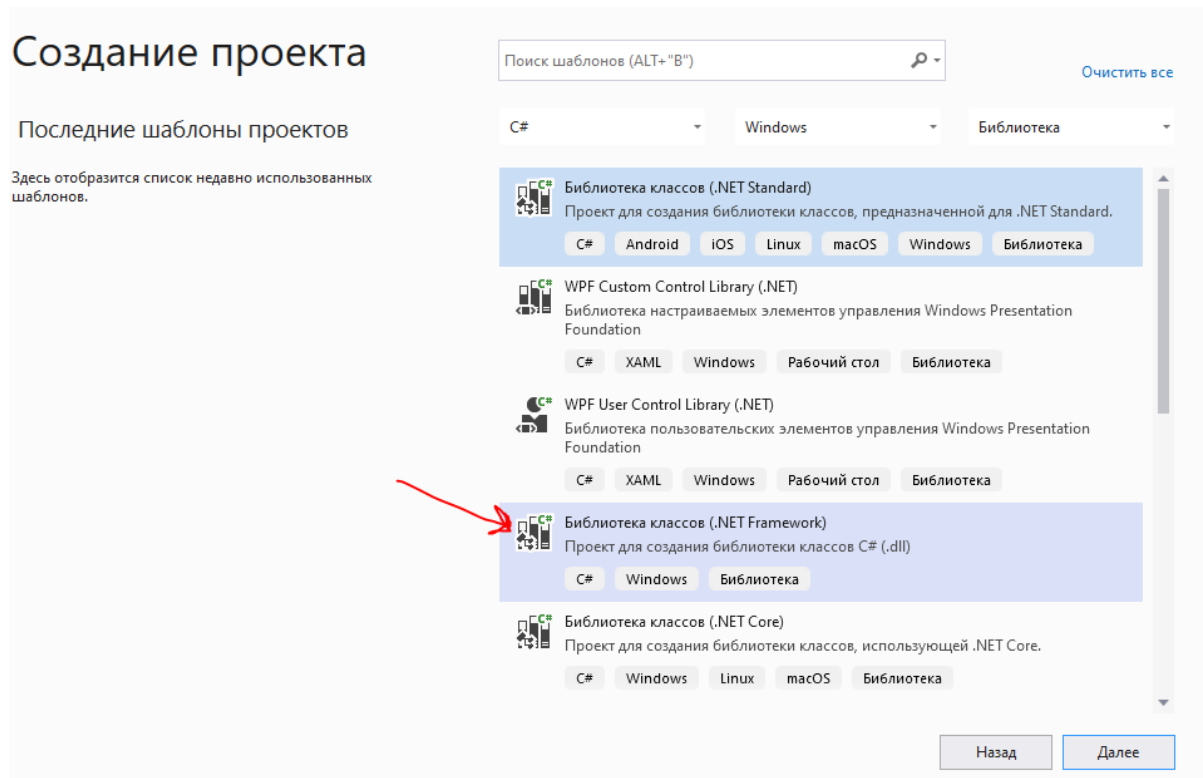


Рисунок 21

## 3. Написать модуль класса

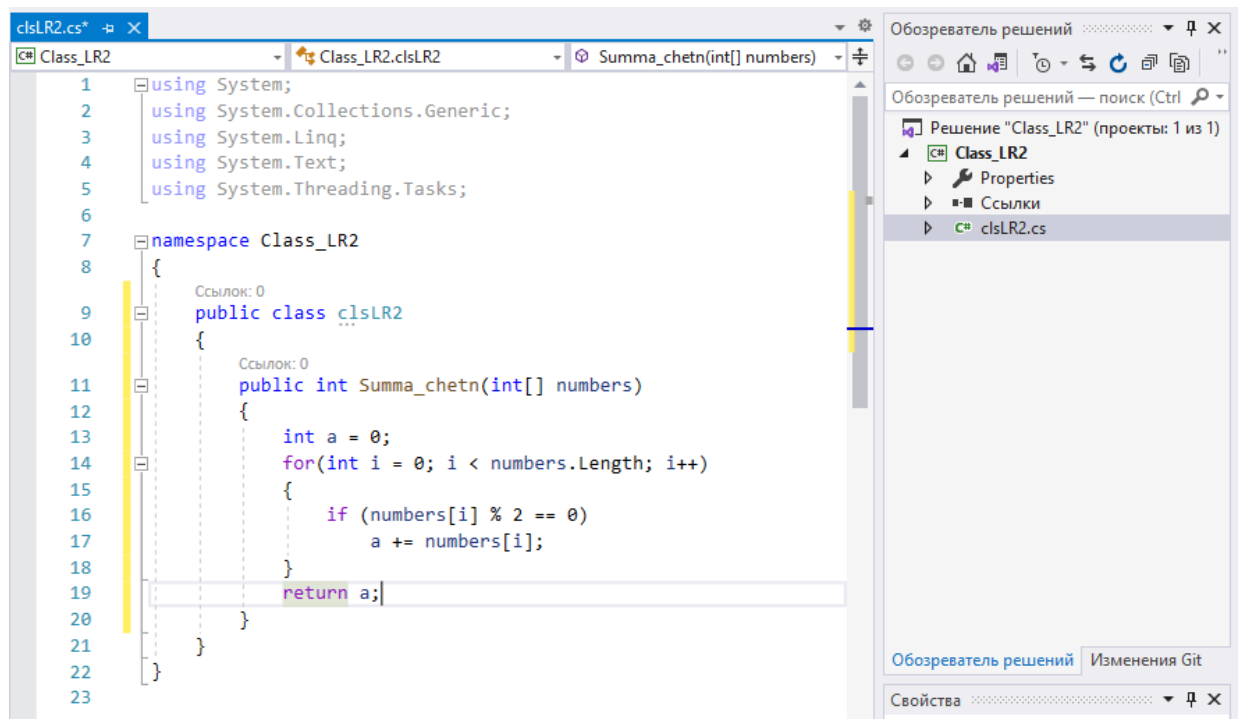


Рисунок 22



#### 4. Запустить IntelliTest из контекстного меню класса (для всех методов)

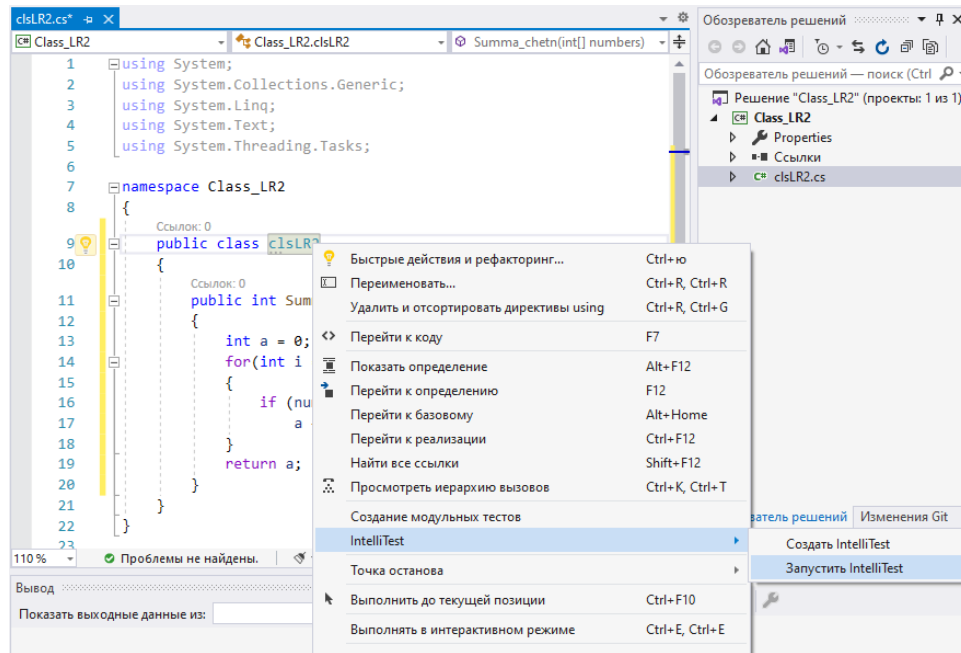


Рисунок 23

#### 5. Изучить и проанализировать результаты тестирования

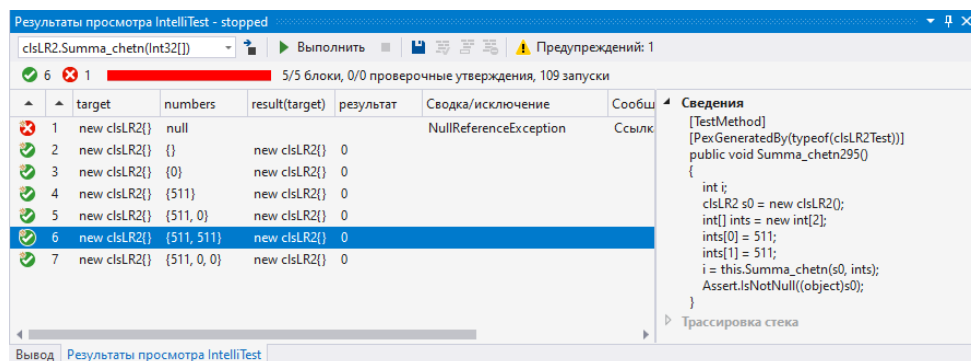


Рисунок 24

#### 6. Сохранить результаты тестирования

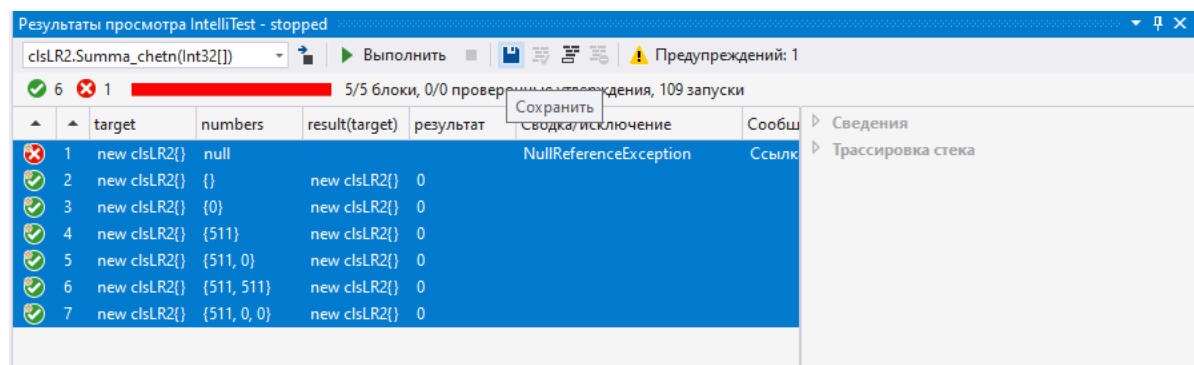


Рисунок 25

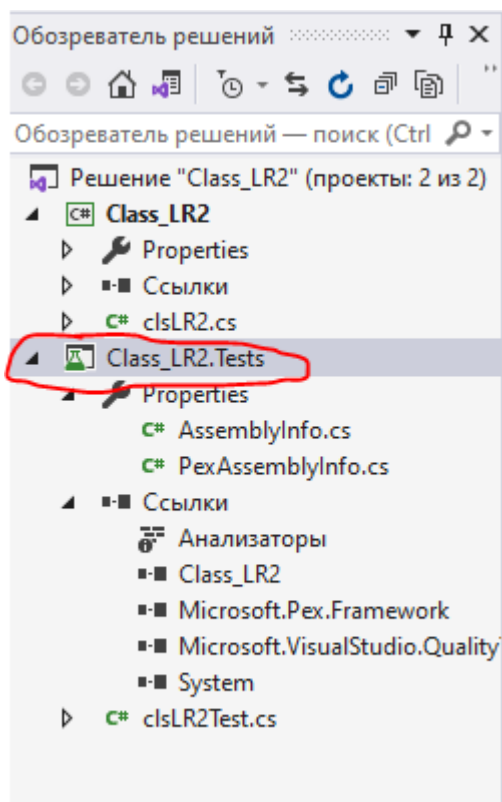


Рисунок 26

## 7. Оформить и сдать отчет о выполненной работе

### Литература

1. Федорова Г.Н. Разработка, внедрение и адаптация программного обеспечения отраслевой направленности: учебное пособие для учебных заведений, реализующих программу среднего профессионального образования, ПМ "Разработка, внедрение и адаптация программного обеспечения отраслевой направленности) / Г. Н. Федорова. - Москва : КУРС: ИНФРА-М, 2019.
2. Гвоздева, Валентина Александровна. Введение в специальность программиста: учебник для студентов образовательных учреждений СПО / В. А. Гвоздева. - Изд. 2-е, испр. и доп. - Москва : ФОРУМ: ИНФРА-М, 2017.
3. Гагарина, Лариса Геннадьевна. Технология разработки программного обеспечения: учебное пособие для студентов вузов/ Л.Г. Гагарина [и др.]; под ред. Л. Г. Гагариной. - Москва : ФОРУМ: ИНФРА-М, 2018.